Universidad
Carlos III de Madrid
www.uc3m.es

Diseño e implementación de una ayuda electrónica para pacientes con baja visión periférica (II)

Autor: Rubén Núñez Martín

Titulación: Grado en Ingeniería Electrónica Industrial y Automática

Profesor Tutor: Juan Carlos Torres Zafra (Departamento de Tecnología Electrónica)

Codirector: Ricardo Vergaz Benito (Departamento de Tecnología Electrónica)

Fecha: 8 de Julio de 2015

Universidad
Carlos III de Madrid
www.uc3m.es

Título: Diseño e implementación de una ayuda electrónica para pacientes con baja visión periférica (II)

Autor: Rubén Núñez Martín

Tutor: Juan Carlos Torres Zafra

Codirector: Ricardo Vergaz Benito

EL TRIBUNAL

Presidente: Virginia Urruchi del Pozo

Vocal: Plinio Jesús Pinzón Castillo

Secretario: Jonathan Crespo Herrero

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día 8 de Julio de 2015 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

VOCAL

SECRETARIO PRESIDENTE

Índice

Índice De Figuras

Agradecimientos

Tras cuatro años de duro trabajo, parece que se acerca el final de un camino que parecía no terminar jamás. Han sido cuatro años en los que he vivido situaciones de todo tipo, tanto a nivel académico como a nivel personal. Situaciones que te animan a seguir para adelante y de las que normalmente tendemos a olvidarnos, y situaciones de las que es complicado salir y que muy difícilmente se pueden sacar de la mente, pero al fin y al cabo situaciones que te hacen aprender.

Llegando al final de esta etapa de mi vida, me siento en la necesidad de agradecer cada situación vivida, las buenas por ayudarme a seguir para adelante, y las malas por enseñarme algo en la vida. Por supuesto me siento en la necesidad de agradecer la ayuda recibida a todas y cada una de las personas que han hecho posible que este sueño se haga realidad.

En primer lugar quiero dar las gracias a mi familia. A mis padres por todo el apoyo recibido durante todos estos años, partiendo de los ánimos y la compañía aportada tanto en los mejores momentos como en los peores, que a pesar de todo no hay mejor compañía que la de tu propia familia. A mi hermano, por el cable que me ha echado en algunos momentos de la carrera y, por supuesto, por sus chistes malos de programador que me animaban en momentos duros. Y, por supuesto, a mi abuelo, una de las personas más importantes en mi vida, que ha sido el que más ha sufrido mis cabreos y mis malos humores y que, aun así, siempre le tengo ahí sea para lo que sea.

A mi novia, Vanessa, por todo el apoyo que me ha dado durante todo este tiempo. Ha sido el pilar fundamental en el que me he apoyado y que me ha servido para tirar para adelante y que me ha hecho tener ganas siempre de dar un poco más de mi pensando en el futuro. Además ella ha sido quien me ha dado un hombro en el que llorar en los peores momentos y ha sido quien ha estado a mi lado en cada celebración. Además también agradecer siempre el interés mostrado en todo lo que yo tenía entre manos. Y por supuesto agradecer que a pesar de todo, siempre me está esperando con una sonrisa en la cara. Sin ella nada de esto hubiera sido posible.

A todos esos compañeros que he tenido durante la carrera y que han hecho que cada clase, cada laboratorio, cada momento en la universidad en general, haya sido mucho más llevadero y mucho más fácil de aguantar, además de esas horas estudiando juntos y ayudándonos a resolver ejercicios unos a otros. En especial me gustaría agradecérselo a David, por todos los momentos vividos, los ánimos aportados y por supuesto la ayuda ante cualquier problema surgido. Y, por supuesto, agradecerle estos cuatro años a mi compañero de proyecto, Jonathan, por todo lo vivido tanto dentro como fuera de la universidad, ya que sin él nada habría sido lo mismo.

No se me puede olvidar mencionar a los amigos que siempre han estado ahí para sacarme una sonrisa cuando no tenía ganas, los mismos que me han acompañado esas noches de cervezas interminables.

Agradecer también a todo el personal del CMRF y, en especial, a todo el personal del IOBA por la gran acogida que nos dieron a mi compañero y a mí en las visitas a estos centros y por toda la ayuda aportada.

También quiero agradecer toda la ayuda aportada durante el último año de carrera por parte de Agus, tanto la ayuda aportada con el desarrollo del TFG como en mi colaboración con el proyecto “Light Access”, así como la relación que se está estableciendo durante todo este tiempo y que espero que siga así mucho tiempo.

Gracias también a todo el grupo GDAF por toda la ayuda aportada durante la carrera, y sobre todo a Ricardo y a Juan Carlos por este último año, en el que la relación ha sido mucho más cercana.

Por último quiero dar las gracias a todas aquellas personas que se me haya olvidado mencionar, que seguro que son varias, pero resulta imposible acordarse de todas y cada una de las personas en este momento.

Resumen

Este proyecto se basa en el desarrollo de una herramienta de ayuda asistencial para aquellas personas que sufren problemas de baja visión.

Se trata de un proyecto desarrollado en completa colaboración con el Instituto de OftalmoBiología Aplicada de la Universidad de Valladolid (IOBA), tratando de cubrir las necesidades que allí nos planteaban, conociendo las necesidades de los usuarios finales.

El proyecto consta de dos partes, una primera en la que se realizará un estudio de los contornos de la imagen capturada por las cámaras para posteriormente situarla en la zona de la visión que el paciente es capaz de ver. Por otro lado, se desarrollará una herramienta para aquellas personas que no son capaces de detectar profundidades.

Este Trabajo Fin de Grado se centra en el desarrollo de la segunda parte, en la que se busca mostrar, mediante contornos de distintos colores, los objetos más cercanos y los más lejanos. Para conseguir este propósito debemos conseguir realizar una calibración de dos cámaras, de modo que las capturas de las mismas estén perfectamente alineadas, permitiéndonos posteriormente la implementación del mapa de disparidad con el que conseguiremos el objetivo deseado.

Abstract

This project is based on the development of a helping tool for those people that suffer low-vision problems.

It is a project developed in complete collaboration with the Applied Ophthalmobiology Institute (IOBA), trying to cover the needs they posed us, knowing the needs of the final users.

The project consist of two parts, the first in which we are going to do a contour's study of the captured image by the cameras to situate it in the vision zone where the patient is able to see. On the other hand, we are going to develop a tool for those people that are not able to detect the depth.

This part of the project is focused on the development of the second part, in which it seeks to show with contours of different colors, the closest and the most distant objects. To get this purpose, we need to get a calibration of two cameras, so that the captures of the cameras are perfectly aligned, allowing us to make the implementation of the disparity map with which we reach the desire target.

Capítulo 1: Introducción Y Objetivos

En este primer capítulo, se va a realizar una introducción a lo que es el diseño para todos, cuyas pautas son las que guían todo el trabajo realizado, para posteriormente hablar sobre la baja visión y las ayudas existentes actualmente. Finalmente se contarán los objetivos del proyecto, además de los requerimientos del mismo y las distintas fases en las que se divide el proyecto.

Hay que destacar antes de comenzar la memoria de este Trabajo Fin de Grado (TFG), que este ha sido realizado conjuntamente con Jonathan Pajares Redondo, por lo que habrá partes comunes en ambos documentos.

1.1. Introducción

El sistema que se va a implementar en este proyecto, trata de conseguir la representación de la escena capturada a partir de unas cámaras mediante sus contornos, haciendo una clara distinción entre aquellos objetos más cercanos y los que estén situados a mayor distancia. Para conseguir esta representación, será necesaria la representación del mapa de disparidad.

Un mapa de disparidad es una representación espacial de la información de profundidad de la escena capturada por las cámaras. Se trata de una representación en escala de grises en la que los objetos más próximos al primer plano tendrán colores más cercanos al blanco, mientras que los objetos más alejados tendrás colores cercanos al negro. Para ello se necesitan dos cámaras para conseguir la conocida como visión estereoscópica.

La visión estereoscópica es la capacidad que tiene un ser vivo de integrar las dos imágenes que está viendo (una por cada ojo) en una sola por medio del cerebro. En nuestro caso, tendremos dos cámaras que actuarán como ojos y un sistema de procesamiento que actuará como cerebro de nuestro sistema.

El desarrollo del proyecto se ha llevado a cabo en colaboración directa con el Instituto de OftalmoBiología Aplicada de la Universidad de Valladolid (IOBA), de donde se han ido sacando ciertas pautas para el desarrollo del proyecto, a partir del trato directo con especialistas que conocen de primera mano las necesidades de los pacientes.

1.1.1. Diseño Para Todos

Somos muchos millones de personas los que habitamos el planeta. Un elevado número de nosotros padecemos dificultades para desarrollar las actividades de la vida diaria, ya sea por discapacidad física o psicológica, discapacidad visual, cualquier tipo de lesión física que dificulte nuestra vida o simplemente algún momento concreto en nuestras vidas en el que nos encontramos con menos facultades de las habituales.

Además, por supuesto todos nosotros vamos envejeciendo con el tiempo, lo que provoca la pérdida de algunas de nuestras capacidades y la disminución de otras. Unas personas sufren estos efectos adversos en edades más tempranas y otros en edades más avanzadas, pero todos nosotros vamos a pasar por esta etapa de dificultades.

El ser humano ha sido capaz de confeccionar el entorno para adaptarlo a sus necesidades, haciendo que algunas funciones resulten más sencillas de llevar a cabo. Sin embargo hay muchas otras que no se han adecuado a las necesidades de las personas y que todos, debido a los problemas mencionados anteriormente, querríamos tener adaptadas.

Pero esto no es siempre posible, dado que hay algunas cosas del entorno que no podemos cambiar, al menos por el momento. Además hay otros factores que no facilitan el desarrollo de los sistemas necesarios para facilitar esas actividades, tales como que en muchas ocasiones los empresarios no se preocupen de prestar atención a las necesidades de los potenciales usuarios, sino que tan solo buscan su propio beneficio.

Por lo tanto, la necesidad de que el entorno se adapte cada vez más a nuestras necesidades, crea la necesidad de un cambio en la concepción de los productos y servicios de manera que estén *diseñados para todos*.

El Diseño para todos busca la adecuación de los productos y los servicios teniendo en cuenta varios factores:

- El entorno debe adecuarse a las necesidades del ser humano y no al revés, ya que el ser humano es capaz de modificar el entorno en busca de sus necesidades.
- Hay una gran variedad de potenciales usuarios cuyas necesidades son muy diferentes debido a la gran diversidad de enfermedades, culturas, etc.
- Con el paso del tiempo y el consecuente aumento en la edad de las personas, las capacidades de las mismas se va deteriorando y, por tanto, las necesidades de dichas personas sufren variaciones muy diversas.
- Conocer las necesidades de los potenciales usuarios, dado que son los que van a necesitar estos productos y servicios, siendo necesario contar con ellos para conocer de primera mano estas necesidades.

Por tanto, se podría definir el Diseño para Todos como una metodología que trata de modificar tanto el entorno, como los productos y servicios necesarios en la vida cotidiana con el objetivo de facilitar las labores diarias a todas las personas y aportar el mayor nivel de independencia posible, buscando con esto el objetivo de construir una sociedad en la que todos los miembros tengan igualdad de oportunidades.

Principios del Diseño para Todos

El Diseño para Todos se basa en siete principios básicos [1]:

1. Uso equitativo: el diseño ha de ser fácil de utilizar para todas las personas, independientemente de sus capacidades o habilidades.
2. Flexibilidad: el diseño tiene que tener versatilidad para adaptarse a una amplia gama de capacidades individuales.
3. Simple e intuitivo: el funcionamiento del sistema debe ser fácil de entender para cualquier usuario.
4. Información comprensible: el diseño tiene que ser capaz de transmitir la información a los usuarios de un modo sencillo de entender.
5. Tolerante a errores: se deben minimizar los posibles accidentes con el sistema para evitar consecuencias no deseadas.
6. Escaso esfuerzo físico: el diseño debe tener una amplia usabilidad con un esfuerzo mínimo.
7. Dimensiones apropiadas: los tamaños de los objetos y los espacios que estos ocupan, han de ser los apropiados para facilitar su usabilidad por parte de cualquier tipo de usuario.

1.1.2. Definición De Baja Visión

El término Baja Visión hace referencia a una anomalía visual que provoca una pérdida en la capacidad visual necesaria para realizar las tareas de la vida diaria. Funcionalmente podríamos definir a una persona con Baja Visión como aquella que es capaz de utilizar la poca cantidad de luz que puede ver para orientarse y emplear la misma con fines funcionales.

Se puede considerar que una persona tiene Baja Visión cuando no es posible corregir dicha visión con gafas, lentes de contacto o cualquier método de tratamiento médico o quirúrgico. Lo habitual para una persona con Baja Visión es realizar procesos de rehabilitación visual y el empleo de diferentes ayudas ópticas para mejorar su calidad de vida.

La OMS (Organización Mundial de la Salud) definió, en 1992, que una persona con Baja Visión es aquella que aun después de tratamiento médico y/o corrección óptica común, tiene una agudeza visual[1] de 0.3 en el mejor ojo o un campo visual inferior a 10^{o} desde el punto de fijación, y que quiere utilizar su visión para la planificación y ejecución de tareas [2].

Actualmente en el mundo hay más de 100 millones de personas que padecen Baja Visión. Se trata de una enfermedad que no conoce diferencias de edad o diferencias económicas, sociales o étnicas sino que afecta a todo el mundo por igual, aunque hay algunas enfermedades que en el tercer mundo no tienen solución mientras que en el primer mundo sí pueden tenerla. El mayor porcentaje de personas afectadas lo encontramos a partir de los 45 años, aunque las personas con una edad por debajo de ésta, no están exentos de padecerla. Sin embargo en el rango de edad comprendido entre los 45 y los 75 años, una de cada seis personas padece estos problemas mientras si miramos en edades superiores a los 75 años, podríamos encontrarnos a una de cada cuatro personas con Baja Visión.

Hay que matizar que la Baja Visión no es solo causa de la edad, sino que también puede ser causada por otros factores como pueden ser traumatismos o enfermedades congénitas. Los problemas más frecuentes son los relativos a la refracción de la luz, aunque existen otros bastante comunes que pueden ser causa de algún problema crónico o funcional que provoque un daño a la visión o cualquier tipo de enfermedad que afecte al ojo, entre los que hay que destacar las enfermedades cerebrales.

[1] Agudeza visual: capacidad de un sistema de visión para percibir, detectar o identificar objetos con unas condiciones de iluminación buenas.

Algunas de las patologías más comunes son:

-MIOPÍA [3]

La miopía es un defecto de refracción del ojo en el cual los rayos de luz paralelos convergen en un punto focal situado delante de la retina, en lugar de converger en la misma retina, lo que provoca que la persona afectada no pueda enfocar de manera correcta los objetos lejanos como observamos en la figura 1, lo que puede provocar déficit de agudeza visual, dolores de cabeza, incomodidad visual o irritación del ojo.

Ocurre cuando el globo ocular es demasiado largo y evita que la luz entrante se enfoque directamente sobre la retina. También puede ocurrir cuando la córnea o el cristalino tienen forma anormal. Este problema suele ser corregido con el uso de lentes divergentes o incluso es muy habitual actualmente recurrir a una cirugía refractiva para mejorar la visión de manera permanente.

Figura 1: Visión con miopía [4]

-HIPERMETROPÍA [5]

La hipermetropía es un defecto de refracción en el que los rayos de luz paralelos convergen en un punto focal detrás de la retina, en vez de converger en la misma retina, lo que provoca que la persona enfoque mejor los objetos lejanos que los que están más cerca (como podemos ver en la figura 2), provocando déficit de agudeza visual, dolores de cabeza, incomodidad visual o irritación del ojo.

Ocurre cuando el globo ocular es demasiado corto, lo que evita que la luz que entra en el ojo se enfoque directamente sobre la retina. También puede ocurrir cuando la córnea o el cristalino tienen forma anormal. Para solucionar este problema se pueden utilizar lentes divergentes, o como ocurre con la miopía, ser sometido a una cirugía refractiva.

Figura 2: Visión con hipermetropía [6]

-ASTIGMATISMO [7]

El astigmatismo es un problema de refracción. Es un trastorno en que el ojo no enfoca la luz de forma pareja sobre la retina.

El astigmatismo ocurre cuando la luz se desvía de manera diferente en función del lugar donde impacte con la córnea, y pasa a través del globo ocular. Un ojo con astigmatismo tiene una córnea con áreas más inclinadas o más redondeadas que otras. Esto puede causar que las imágenes se vean borrosas o alargadas (véase figura 3) provocando déficit de agudeza visual, dolores de cabeza, incomodidad visual, irritación del ojo o dificultad para manejarse por las noches.

Este problema suele ser corregido mediante el uso de lentes convergentes, aunque como ocurre con la miopía y la hipermetropía, la cirugía es una solución muy utilizada.

Figura 3: Visión con astigmatismo [8]

-DMAE (DEGENERACIÓN MACULAR ASOCIADA A LA EDAD) [9]

La DMAE o Degeneración Macular Asociada a la Edad es una enfermedad degenerativa de la zona central de la retina, o mácula, que provoca un deterioro progresivo de las células y del epitelio pigmentario de la retina. Como consecuencia, se produce una pérdida de la visión central, necesaria para realizar las actividades en las que hay que ver directamente hacia delante (véase figura 4). La degeneración macular no causa dolor.

En algunos casos, la degeneración macular relacionada con la edad avanza tan lentamente que las personas no notan cambio alguno en su visión. En otros casos, la enfermedad progresa rápidamente y puede causar una pérdida de la visión en ambos ojos. Hay varios métodos de tratamiento tales como el tratamiento con una formulación con altas dosis de antioxidantes y zinc, cirugía láser, terapia fotodinámica o diversos tipos de inyecciones.

Figura 4: Visión con DMAE [10]

-GLAUCOMA [11]

El glaucoma es una enfermedad que pueden dañar al nervio óptico del ojo. Por lo general no presenta síntomas y puede provocar la pérdida de la visión de manera repentina, sin embargo, con exámenes oftalmológicos periódicos, la detección temprana y el tratamiento puede preservarse la vista.

En la parte delantera del ojo existe un espacio, la cámara anterior, en el que entra y sale un líquido continuamente alimentando los tejidos de su alrededor. El líquido sale de la cámara anterior a través del ángulo abierto donde se unen la córnea y el iris. Cuando el líquido llega al ángulo fluye a través de una red o malla esponjosa y sale del ojo. A veces, cuando el líquido llega a este ángulo, pasa lentamente a través de esta malla esponjosa y, al acumularse, la presión dentro del ojo aumenta hasta llegar a un nivel en que puede dañar al nervio óptico. Cuando esto ocurre se puede provocar un glaucoma de ángulo abierto. Una persona con glaucoma padece lo que se conoce como visión en túnel, tal como se muestra en la figura 5.

Esta enfermedad tiene diversos tipos de tratamiento, cuya efectividad depende de lo que se tarde en diagnosticar el glaucoma. Entre los tratamientos más utilizados, destacan el consumo de medicamentos, la trabeculoplastia o cirugía con láser y la cirugía convencional.

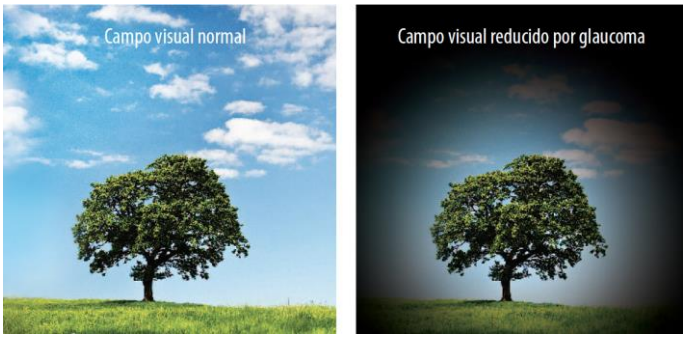


Figura 5: Visión con glaucoma [12]

-CATARATAS [13]

Una catarata es una nubosidad (opacidad) en el cristalino que dificulta la visión. El cristalino es la lente interior del ojo que sirve para enfocar y que de normal es clara y transparente. No se trata de ninguna enfermedad sino de un envejecimiento ocular.

Padecer cataratas provoca una visión borrosa, una percepción distorsionada de los colores, mayor sensibilidad a destellos de luz, mala visión en ambientes oscuros, visión doble y necesidad de cambios frecuentes en las gafas o las lentes de contacto, como podemos observar en la figura 6.

Los principales tratamientos para este problema, es el uso de gafas, mejor iluminación, gafas anti-reflectoras para el sol o lentes de aumento. En el momento que estas medidas dejan de funcionar, la única solución posible es la cirugía.

Figura 6: Visión con cataratas [14]

-RETINOPATÍA DIABÉTICA [15]

La retinopatía diabética es una complicación de la diabetes y una de las causas principales de la ceguera actualmente. Ocurre cuando la diabetes daña a los pequeños vasos sanguíneos de la retina, un tejido sensible a la luz situado en la parte posterior del ojo.

Una persona que padezca retinopatía diabética, al principio no notará ningún cambio en su visión. Pero con el tiempo, la retinopatía diabética puede empeorar y causar una pérdida en la visión. Normalmente esta enfermedad afecta a ambos ojos (véase figura 7).

Actualmente la única solución para esta enfermedad es una cirugía conocida como fotocoagulación retiniana, con la que se trata de reducir el tamaño de los vasos sanguíneos afectados.

Figura 7: Visión con retinopatía diabética [16]

RETINOSIS PIGMENTARIA [17]

La retinosis pigmentaria es la enfermedad hereditaria más frecuente de la retina. Se caracteriza por una degeneración progresiva de la retina. Actualmente carece de tratamiento, y su gravedad hace que sea una de las patologías oculares de origen genético sobre las que más se está investigando. La terapia génica está ofreciendo resultados muy esperanzadores.

Según se desprende de las conclusiones del Congreso internacional de Retina celebrado en el Instituto de Microcirugía Ocular (IMO) los días 8 y 9 de junio de 2013 [18], estudios en fase I en humanos ya parecen estar demostrando cómo el uso de células madre para reemplazar células dañadas de la retina logra mejorar la agudeza visual de los pacientes. Esta terapia se aplica en pacientes que pierden células fotorreceptoras y/o del epitelio pigmentario, un tipo de células que no se regeneran, y que son fundamentales para la visión.

Figura 8: Visión con retinosis pigmentaria [19]

1.1.3. Ayudas Ópticas Prescritas En La Consulta

Cada vez es más elevado el número de especialistas en el campo de la visión, encontrando diversas funcionalidades entre las que destacan la oftalmología, la rehabilitación visual y la optometría. Cuando un oftalmólogo considera inapropiada una intervención quirúrgica para subsanar un problema visual, o cuando se han acabado todas las alternativas para ello, el paciente es considerado como una persona con Baja Visión y ha de ser tratado en un centro especializado, en el que entran en juego tanto la función del optometrista como la del rehabilitador visual.

El optometrista se encarga de evaluar los restos visuales del paciente y marcar las ayudas necesarias para cada situación, mientras el rehabilitador visual tiene la función de ayudar al paciente a familiarizarse con las ayudas establecidas y enseñarle a utilizar de modo correcto dichas ayudas para optimizar la utilización de las mismas. El rehabilitador visual ha de tener siempre en mente que cada persona tiene un problema diferente, así como una personalidad distinta, por lo que debe ajustar el programa de rehabilitación a la persona afectada.

Podríamos diferenciar cuatro tipos de ayudas para pacientes con Baja Visión: [20]

- Ayudas ópticas: se trata de ayudas manuales, o montadas en gafas, que proporcionan el aumento necesario según las necesidades de cada paciente. Entre ellas encontramos algunas como lupas, telescopios…
- Ayudas no ópticas: son aquellas medidas que se pueden adoptar para mejorar la postura, el contraste, el deslumbramiento o la distancia de trabajo, con el fin de aumentar la calidad de vida del paciente. Entre ellas encontramos atriles, sistemas de iluminación…
- Ayudas electro-ópticas: aparatos de nueva tecnología que facilitarán al paciente la realización de las tareas de la vida cotidiana, como puede ser el caso de la utilización de lupas electrónicas para la ayuda a la lectura.
- Ayudas no visuales: permiten mejorar la autonomía del paciente sin que la visión del mismo intervenga. Algunos dispositivos catalogables en este grupo pueden ser relojes parlantes, monederos especiales…

Como hemos podido observar hay una gran cantidad de ayudas para personas afectadas por la baja visión, entre las que hemos comentado las ayudas ópticas, no ópticas, electro-ópticas y no visuales.

Sin embargo, dentro de la baja visión hay un grupo de personas que padecen lo conocido como baja visión periférica, es decir, personas que han perdido parte de su campo visual. Para este tipo de personas, el número de ayudas disponibles en la actualidad es mucho más reducido, encontrando algunas tales como los correctores de campo o el sistema Jordy, aunque también son útiles las ayudas mencionadas anteriormente.

- **Correctores de Campo**: lentes utilizadas para corregir algunos tipos de deficiencias visuales. Estos sistemas permites recolocar la visión del paciente o incluso aumentar su campo visual, permitiendo a los pacientes ver una parte del entorno que antes era negra para ellos (ver figura 9).
- **Sistema Jordy**: dispositivo electrónico que permite a los usuarios utilizar de manera eficiente el resto visual del que disponen mediante la magnificación de imágenes y la variación del contraste y los colores de las imágenes, obteniendo una sensación similar a la realidad virtual, (ver figura 10).

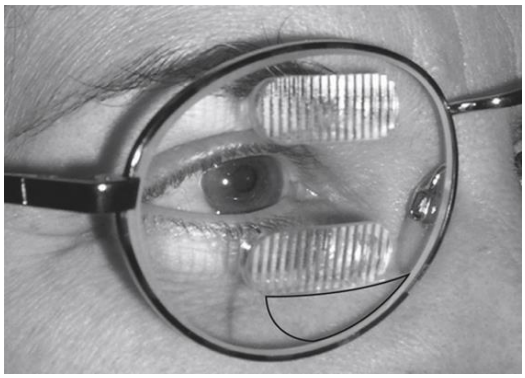


Figura 10: Corrector de campo [21]

Figura 9: Sistema Jordy

Además de estas ayudas ya existentes, hay otras muchas en pleno desarrollo tales como gafas con filtros especiales, lupas con características concretas, etc.

1.2. Objetivos

El objetivo principal del proyecto consiste en tratar de conseguir una herramienta optimizada y operativa, que sirva de ayuda a personas que tengan ciertos tipos de restricciones en el campo visual.

Cuando se sufren ciertas enfermedades como puede ser el caso de un glaucoma, retinosis pigmentaria o cataratas, el paciente puede perder parte de su visión, produciéndose diversos problemas visuales tales como el conocido efecto túnel o manchas distribuidas por todo el campo visual (como vemos en las figuras 1 a 8), lo que puede provocar una gran pérdida de información que lleve al desarrollo de ciertos temores basados en las dificultades de movilidad, principalmente, y por tanto una cierta dependencia en los pacientes que las sufren.

Este proyecto surge como la necesidad de diseñar un sistema de ayuda visual para pacientes con estos síntomas, y ofrecerles la oportunidad de poder llevar una vida completamente normal, sin necesidad de ayuda de otra persona física, es decir, se busca dar una cierta autonomía a pacientes que hayan perdido la misma como consecuencia de ciertas enfermedades de la visión.

Este trabajo se centra en el desarrollo de un software para utilizar en gafas de realidad aumentada, que facilitará al usuario la identificación de los objetos presentes en una escena. Se buscará la detección de los contornos de dichos objetos con el fin de aportar tan solo la información necesaria. Además de mostrar al paciente los objetos de la escena en forma de contorno, estos contornos tendrán un color u otro (verde, azul o rojo) en función de la distancia del objeto a la persona que porte el sistema.

Como base del proyecto se va a partir de PFC de compañeros anteriores tales como el de Carlos Barranco[22] o el de Francisco Collado[23], por lo que se van a utilizar más o menos los mismos instrumentos utilizados en dichos proyectos, como pueden ser unas gafas con dos micropantallas integradas que, además, tienen incorporadas las cámaras de adquisición.

Por tanto, podemos concluir que el objetivo del proyecto es desarrollar una aplicación que permita captar dos imágenes y superponerlas para conseguir la detección de distancias y profundidades y mostrar esta unión de imágenes en unas gafas con dos micropantallas.

Además, unido a este objetivo, nos hallamos ante la necesidad de transformar la implementación de los algoritmos desarrollados previamente, en los TFG de Carlos Barranco y Francisco Collado, a un lenguaje de programación más versátil, para lo que utilizaremos las librerías de *OpenCV* que se explicarán más adelante, aportándonos con ello la posibilidad de implementar dichas herramientas en sistemas de bajo coste y con mayor simplicidad.

1.3. Especificaciones Del Sistema

El sistema que se va a desarrollar en este proyecto tiene unos requerimientos básicos que ha de cumplir para satisfacer las necesidades de los potenciales usuarios.

En primer lugar hay que tener en cuenta que debemos de tener un sistema completamente portable y utilizable en cualquier lugar, por lo que también resulta un factor crítico la autonomía del sistema completo.

Además ha de ser un sistema sencillo que pueda utilizar cualquier persona, así como un precio reducido que pueda afrontar una persona que lo necesite.

Otro de los requerimientos que se deberían cumplir, que no es importante a nivel técnico pero si a nivel personal de los usuarios, es la presencia del sistema, es decir, debe ser un sistema discreto que pase lo más desapercibido posible.

El sistema estará compuesto por tres elementos diferentes que se unirán para dar lugar al elemento definitivo: cámaras, pantallas y un sistema de procesamiento (ver figura 11).

Figura 11: Esquema de los elementos

- **Sistema de adquisición de imágenes**

Nuestro sistema se basa en la visión estereoscópica. La visión estereoscópica es la facultad que tiene un ser vivo de integrar las dos imágenes que está viendo (una por cada ojo) por medio del cerebro.

El primer elemento de nuestro sistema va a ser el sistema de adquisición de imágenes, tratando de simular la visión estereoscópica mencionada anteriormente. Para ello necesitamos dos cámaras con cualidades similares que han de estar perfectamente alineadas para conseguir el efecto deseado.

La idea principal es la de incorporar la cámara elegida en unas gafas, de modo que simulen la visión que pueden aportar los ojos de una persona. Nuestra cámara llevará una conexión directa al sistema de procesamiento, de modo que no será necesario ningún sistema de alimentación para este dispositivo.

- **Sistema de procesamiento**

El principal objetivo del proyecto, es conseguir una ayuda de bajo coste para pacientes con baja visión. Este elemento de nuestro sistema es clave en este aspecto, dado que puede tener un coste muy elevado. Se han barajado numerosas alternativas como podría ser un mini-PC convencional, se han estado valorando varios modelos de la marca NVIDIA, sin embargo no encajan en el objetivo del bajo coste, lo que nos ha llevado a la elección de una RaspberryPi 2.

RaspberryPi es un ordenador de tamaño reducido con un procesador ARM Cortex-A7 de cuatro núcleos de origen británico. Es un computador de coste muy reducido, se puede encontrar por un precio de aproximadamente 30€, lo que encaja perfectamente con los requerimientos del proyecto.

La alimentación de este dispositivo ha de ser mediante baterías o pilas para evitar una conexión a la red y dotar así al sistema de una cierta versatilidad y autonomía, ya que nuestro sistema requiere de su utilización en cualquier lugar, no sirviendo como alimentación una conexión a la red eléctrica.

- **Pantallas**

Como se comentó en el elemento anterior, la idea del proyecto es integrar todos los elementos en unas gafas, por lo que el lugar donde realizaremos la representación de las imágenes capturadas se llevará a cabo en dos micropantallas que forman parte de las propias gafas.

Se trata de dos displays que utilizan conexión USB. Las pantallas, al igual que ocurría con las cámaras, irán conectadas directamente al sistema de procesamiento, que será el encargado de llevar a cabo la alimentación de las mismas.

- **Algoritmos**

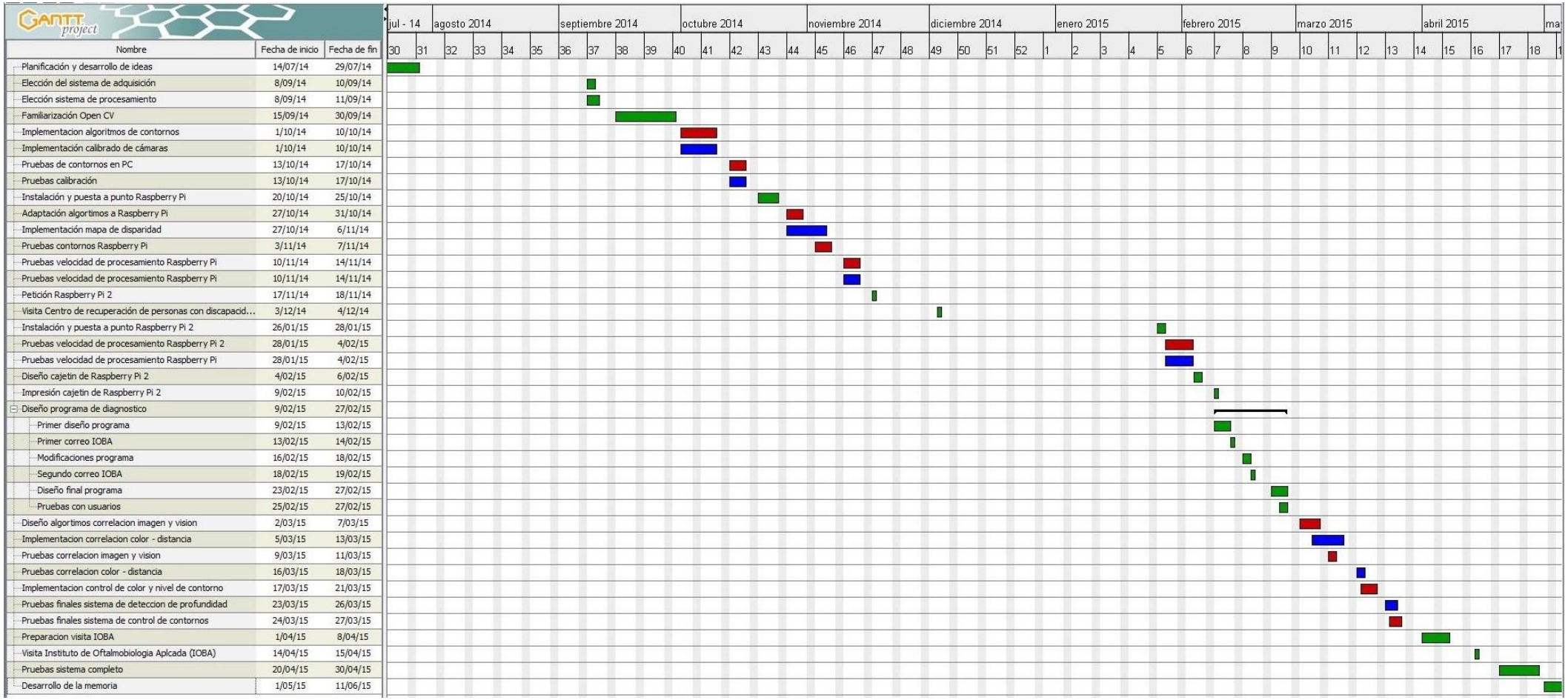
Los algoritmos desarrollados para este proyecto son muy diversos. Desde algunos que resultan necesarios para la calibración de las dos cámaras, de modo que se puedan superponer las imágenes de ambas cámaras y conseguir así la visión estereoscópica, hasta los algoritmos propios para conseguir tanto la imagen en contornos como los distintos colores en función de la distancia a la que se encuentren los mismos.

Además de esto, se ha desarrollado un programa de diagnóstico, que permite a la persona con problemas de visión poder elegir el lugar de visión óptimo en el que colocar la imagen dentro de las pantallas, con la ayuda del oftalmólogo.

1.4. Fases Del Proyecto

Las fases sufridas a lo largo del desarrollo del proyecto son las que se muestran en la siguiente figura (figura 12) que muestra el diagrama de Gantt de este proceso.

En color verde se representa la parte común a ambos proyectos, mientras que la parte roja corresponde a la detección de contornos y el color azul representa la detección de profundidades.



Nombre	Fecha de inicio	Fecha de fin
Planificación y desarrollo de ideas	14/07/14	29/07/14
Elección del sistema de adquisición	8/09/14	10/09/14
Elección sistema de procesamiento	8/09/14	11/09/14
Familiarización Open CV	15/09/14	30/09/14
Implementacion algoritmos de contornos	1/10/14	10/10/14
Implementación calibrado de cámaras	1/10/14	10/10/14
Pruebas de contornos en PC	13/10/14	17/10/14
Pruebas calibración	13/10/14	17/10/14
Instalación y puesta a punto Raspberry Pi	20/10/14	25/10/14
Adaptación algoritmos a Raspberry Pi	27/10/14	31/10/14
Implementación mapa de disparidad	27/10/14	6/11/14
Pruebas contornos Raspberry Pi	3/11/14	7/11/14
Pruebas velocidad de procesamiento Raspberry Pi	10/11/14	14/11/14
Pruebas velocidad de procesamiento Raspberry Pi	10/11/14	14/11/14
Petición Raspberry Pi 2	17/11/14	18/11/14
Visita Centro de recuperación de personas con discapacid...	3/12/14	4/12/14
Instalación y puesta a punto Raspberry Pi 2	26/01/15	28/01/15
Pruebas velocidad de procesamiento Raspberry Pi 2	28/01/15	4/02/15
Pruebas velocidad de procesamiento Raspberry Pi	28/01/15	4/02/15
Diseño cajetin de Raspberry Pi 2	4/02/15	6/02/15
Impresión cajetin de Raspberry Pi 2	9/02/15	10/02/15
Diseño programa de diagnostico	9/02/15	27/02/15
Primer diseño programa	9/02/15	13/02/15
Primer correo IOBA	13/02/15	14/02/15
Modificaciones programa	16/02/15	18/02/15
Segundo correo IOBA	18/02/15	19/02/15
Diseño final programa	23/02/15	27/02/15
Pruebas con usuarios	25/02/15	27/02/15
Diseño algortimos correlacion imagen y vision	2/03/15	7/03/15
Implementacion correlacion color - distancia	5/03/15	13/03/15
Pruebas correlacion imagen y vision	9/03/15	11/03/15
Pruebas correlacion color - distancia	16/03/15	18/03/15
Implementacion control de color y nivel de contorno	17/03/15	21/03/15
Pruebas finales sistema de deteccion de profundidad	23/03/15	26/03/15
Pruebas finales sistema de control de contornos	24/03/15	27/03/15
Preparacion visita IOBA	1/04/15	8/04/15
Visita Instituto de Oftalmobiologia Aplcada (IOBA)	14/04/15	15/04/15
Pruebas sistema completo	20/04/15	30/04/15
Desarrollo de la memoria	1/05/15	11/06/15

Figura 12: Diagrama de GANTT del proyecto

Capítulo 2: Diseño Del Sistema

En este capítulo se va a realizar diversos estudios de mercado en los cuales se van a mostrar diferentes tecnologías tanto del sistema de adquisición de imágenes como de los sistema de procesamiento y de salida, mostrando cuales son los dispositivos con los que se van a realizar las pruebas del proyecto.

Además se describirá el lenguaje de programación utilizado y las plataformas en las que se realizará dicha programación, así como los motivos por los que se han elegido. También se describirá la función realizada por cada uno de los algoritmos desarrollados y un esquema del sistema completo.

2.1. Sistema De Adquisición De Imágenes

Para el correcto funcionamiento de la aplicación a desarrollar, se tiene la necesidad de capturar imágenes estereoscópicas. Para conseguir este tipo de captura, no es suficiente la utilización de una única cámara, sino que se necesitan dos cámaras que estén perfectamente alineadas y con una distancia similar a la existente entre los dos ojos de una persona. Además, las cámaras han de tener las mismas características para que todo funcione de modo correcto. Adicionalmente se realizará un proceso de calibración por software para asegurarnos del correcto alineamiento de ambas cámaras.

Dada la elevada complejidad que conlleva el conseguir un perfecto alineamiento a nivel físico de las dos cámaras, la opción ha derivado en la búsqueda de cámaras con doble objetivo diseñadas para la captura en tres dimensiones. A continuación se puede observar un pequeño estudio de las diferentes alternativas presentes actualmente en el mercado junto a las características más importantes de las mismas, tales como son los px (píxeles) o los fps (frames por segundo que es capaz de capturar la cámara):

3D 2D Webcam from Thanko

- Conexión: USB2.0
- Resolución: 320x240 px
- Velocidad máxima de captura: 15 fps
- Precio: 145 $

Figura 13: Thanko 3D webcam

- http://usbtips.com/usb-toy-3d-webcam-from-thanko/ (15/06/2015)

König CMP 3D Webcam

- Conexión: USB2.0
- Resolución: 640x480 px
- Velocidad máxima de captura: 60 fps
- Precio: 37.99 $

Figura 14: König 3D webcam

- http://www.amazon.com/Konig-Webcam-Glasses-Powered-CMP-WEBCAM3D10/dp/B004CRYEO0 (15/06/2015)
- http://www.konigelectronic.com/es_es/informatica/video/55821926 (15/06/2015)

Zivora 3D Webcam

- Conexión: USB2.0
- Resolución: 640x480 px
- Velocidad máxima de captura: 60 fps
- Precio: 32 €

Figura 15: Zivora 3D webcam

- http://www.solostocks.com/venta-productos/equipos-audio-video/microfonos/zivora-webcam-3d-5996600 (15/06/2015)

Webcam 3D Crazy Cam

- Conexión: USB
- Resolución: 800x600 px
- Velocidad máxima de captura: 30 fps
- Precio: 89.89 €

Figura 16: Webcam 3D Crazy Cam

- http://urban-factory.com/es/accesorios-informaticos/269-crazy-cam-webcam-3d-6-lunettes-3760170841762.html (15/06/2015)

N3D-02BMA 3D Webcam

- Conexión: USB2.0
- Resolución: 640x480 px
- Velocidad máxima de captura: 30 fps
- Precio: 27 $

Figura 17: N3D-02BMA 3D webcam

- http://factory.dhgate.com/product/3d-usb-2-0-ip-webcamera-p715967.html (15/06/2015)

Creative Senz3D Webcam

- Conexión: USB2.0
- Resolución: 1280x720 px
- Velocidad máxima de captura: 30 fps
- Precio: 99.99€

Figura 18: Creative Senz3D webcam

- http://es.creative.com/p/web-cameras/creative-senz3d (15/06/2015)

Minoru 3D Webcam

- Conexión: USB2.0
- Resolución: 800x600 px
- Velocidad máxima de captura: 30 fps
- Precio: 46.09 €

Figura 19: Minoru 3D webcam

- http://www.minoru3d.com/ (15/06/2015)
- http://www.amazon.es/Minoru-MINORU-3D-Webcam/dp/B001NXDGFY/ref=sr_1_1?ie=UTF8&qid=1434364746&sr=8-1&keywords=webcam+3d (15/06/2015)

En nuestro caso, el parámetro críticos que determina la viabilidad del uso del sistema es la velocidad de captura, ya que tenemos la necesidad de realizar una captura en tiempo real, siempre teniendo en cuenta que el cuello de botella no tiene por qué ser nuestro sistema de adquisición. Para conocer cuál sería una buena velocidad de captura de nuestro sistema de adquisición de imágenes debemos tener en cuenta que, para que el ojo humano no aprecie parpadeo, se requiere una frecuencia de captación de imágenes de 50 Hz [24], de modo que el ojo del usuario sea capaz de observar un flujo continuo, y no saltos de imágenes, al menos debemos conseguir una velocidad de captura de 30fps, misma tasa utilizada en el cine hasta el momento para la reproducción de películas. Sin embargo es necesario comprobar dichos tiempos experimentalmente, ya que no siempre se cumplen las características definidas por el fabricante, ya que este aporta la información para las características más significativas, y no para todas las resoluciones a las que puede capturar la cámara.

En el caso de nuestra aplicación, dado que nuestro sistema está pensado para gente con baja visión que no les permitirá una resolución tan detallada, además de que lo que se va a representar son los contornos de lo que se tiene alrededor, no necesitamos que dichos contornos sean extremadamente finos. Ello rebaja las necesidades de resolución.

Por supuesto, hay que tener en cuenta que la idea para el diseño definitivo pasa por la integración de las cámaras y las gafas (véase Apartado 2.4) en un mismo dispositivo, aunque no es necesariamente un problema, ya que se puede realizar un desmontaje de la estructura de la cámara y utilizar tan solo lo que necesitamos.

Por último, y no por ello menos importante, es la necesidad de realizar un sistema de bajo coste, por lo que nuestro sistema de adquisición de imágenes debe cumplir, además de todo lo anterior, que tenga el menor coste posible.

Para realizar las pruebas del proyecto se ha utilizado la última cámara descrita, la *Minoru 3D Webcam*, incluso desmontándola y acoplándola al sistema de salida mediante un soporte realizado con una impresora 3D, tal como observamos en la figura 20:

Figura 20: Montaje de la cámara

2.2. Sistema De Procesamiento

Para nuestro proyecto necesitamos un sistema de procesamiento que sea totalmente portátil. Dado que necesitamos que se pueda utilizar en cualquier momento y en cualquier lugar, debe ser cómodo de transportar. Además debemos tener en cuenta tanto las necesidades de los programas a utilizar, como la conexión de nuestro sistema de adquisición de imágenes y del sistema de salida, en el que se mostrarán las imágenes después de su tratamiento.

A continuación se muestra un pequeño estudio de los diferentes sistemas de procesamiento con estas características que actualmente están en el mercado. Hay que tener en cuenta que es un campo en continuo desarrollo en el que cada día hay nuevos productos y de mejores prestaciones.

Gigabyte GB-BXBT-2807 Brix - Mini PC

- Dimensiones (mm): 56.1 x 107.6 x 114.4
- Peso: 0.69 kg
- Procesador: Intel Celeron Processor N2807 up to 2.17GHz
- Tarjeta gráfica: Intel HD graphics
- Puertos de conexión:
    - 1 x USB3.0
    - 2 x USB2.0
    - 1 x RJ45,
    - 1 x HDMI
    - 1 x Audio/Micrófono
    - 1 x VGA
- Precio aproximado: 119 €
- http://www.pccomponentes.com/gigabyte_gb_bxbt_2807_brix.html (10/06/2015)

Figura 21: Mini PC Gigabyte

Rikomagic MK12 2GB/16GB Quad Core Android PC - Mini PC

- Dimensiones (mm): 116 x 112 x 18.91
- Peso: 0.6 kg
- Procesador: Amlogic S812,Quad core ARM Cortex-A9r4, 2.0GHz
- Tarjeta gráfica: Octa core ARM Mali-450 GPU up to 600MHz+ (DVFS)
- Puertos de conexión:
    - 1 x HDMI1.4b
    - 2 x USB2.0
    - 1 x USB host OTG
    - 1 x Salida óptica
    - 1 x microSD
    - 1 x RJ45
- Precio aproximado: 90 €
- http://www.pccomponentes.com/rikomagic_mk12_2gb_16gb_quad_core_android_pc.html (10/06/2015)

Figura 22: Mini PC Rikomagic

Asus VivoPC VM42-S081V Intel Celeron 2957U/4GB/500GB Plata - Mini PC

- Dimensiones (mm): 190 x 190 x 56.2
- Peso: 1.2 kg
- Procesador: INTEL Celeron 2957U
- Tarjeta gráfica: Intel HD Integrated Graphics
- Puertos de conexión:
  - 4 x USB3.0
  - 2 x USB2.0
  - 1 x HDMI
  - 1 x Display Port
  - 1 x RJ45
  - 2 x Audio-jacks
  - 1 x S/PDIF
  - 1 x lector de tarjetas
- Precio aproximado: 245 €
- http://www.amazon.es/Asus-VM42-Mini-ordenador-sobremesa-Graphics/dp/B00QLX2G66 (10/06/2015)

Figura 23: Mini PC Asus

Nantec Ebox 2300

- Dimensiones (mm): 115 x 115 x 35
- Peso: 0.5 kg
- Procesador: Vortex86 200 MHz
- Tarjeta gráfica: 2D/3D Graphics con MPEG2
- Puertos de conexión:
  - 1 x RJ45 3 x USB1.1
  - 1 x PS/2
  - 1 x Adaptador IDE para memoria flash
  - 1 x Adaptador CompactFlash
  - 2 x Audiojacks
  - 2 x RS232
  - 1 x 6-pin para teclado
  - 1 x 6-pin para ratón
- Precio aproximado: 50 €
- http://www.amazon.es/Nantec-Ebox-2300-sobremesa-plateado/dp/B00Q8BCNTG (10/06/2015)

Figura 24: Mini PC Nantec

Raspberry Pi Modelo B - Mini PC

- Dimensiones (mm): 85 x 56 x 17
- Peso: 50 g
- Procesador 700 MHz Low Power ARM1176JZFS
- Tarjeta gráfica: Dual Core VideoCore IV
- Puertos de conexión:
  - 2 x USB2.0
  - 1 x HDMI
  - 1 x microUSB
  - 1 x RJ45
  - 1 x Audio-jacks
  - 1 x Ethernet
  - 1 x SD
  - 26 x GPIO
  - 1 x puerto serie para cámara
  - 1 x puerto serie para display

Figura 25: Mini PC Raspberry Pi B

- Precio aproximado: 35 €
- http://www.amazon.es/Raspberry-Pi-RBCA000-Model-1176JZF-S/dp/B008PT4GGC

Raspberry Pi 2 Modelo B - Mini PC

- Dimensiones (mm): 85 x 56 x 17
- Peso: 50 g
- Procesador: Quad-Core ARM Cortex A7 a 900MHZ
- Tarjeta gráfica: Dual Core VideoCore IV Multimedia Co-Processor
- Puertos de conexión:
  - 4 x USB2.0
  - 1 x HDMI
  - 1 x microUSB
  - 1 x RJ45
  - 1 x Audio-jacks
  - 1 x Ethernet
  - 1 x microSD
  - 40 x GPIO
  - 1 x puerto serie para cámara
  - 1 x puerto serie para display

Figura 26: Mini PC Raspberry Pi 2 B

- Precio aproximado: 40 €
- http://www.pccomponentes.com/raspberry_pi_2_modelo_b.html (10/06/2015)

El requisito fundamental para el proyecto, tal y como se ha venido comentando desde el principio del documento, es la portabilidad del sistema y el precio del mismo. Por lo que debemos encontrar un sistema de procesamiento que satisfaga nuestras necesidades de procesamiento al mínimo coste y tamaño posibles.

Hay que tener en cuenta las conexiones disponibles en el sistema de procesamiento, dada la necesidad de conexión tanto del sistema de adquisición de imágenes como el sistema de salida, es decir, hay que tener en cuenta tanto las conexiones USB para las cámaras como la conexión HDMI del sistema de salida. Además, se ha de tener en cuenta que la idea es alimentar dicho dispositivo con una batería portátil, sin necesidad de estar conectado a la red eléctrica, por lo que el consumo es importante.

Para la realización del proyecto se han realizado pruebas en los dos últimos modelos descritos, Raspberry Pi Modelo B y Raspberry Pi 2 Modelo B. Ambos modelos aportan las soluciones necesarias al proyecto, tanto las conexiones HDMI y USB como el bajo coste. Los resultados obtenidos con ambos dispositivos serán comentados más adelante.

2.3. Algoritmos

2.3.1. Lenguaje De Programación

Dada su aplicación, el requerimiento más crítico para nuestro sistema es el funcionamiento del sistema en tiempo real. Para el desarrollo de los programas necesarios, se ha utilizado el lenguaje *Python,* debido a su facilidad de uso y su mayor velocidad respecto a otros lenguajes de programación, además de que el sistema de procesamiento lleva instalado un IDLE (Integrated DeveLopment Environment) o entorno de desarrollo de *Python.*

Además, para el desarrollo de los códigos de programación vamos a apoyarnos en las librerías de *OpenCV*, las cuales nos van a permitir implementar un código mucho más sencillo de entender y mucho más eficaz, mediante el uso de las funciones ya implementadas que tiene dicha librería.

Todo este desarrollo se realizará en *Raspbian,* sistema operativo desarrollado para el dispositivo elegido (*RaspberryPi).*

Raspbian [25] [26]

Raspbian es un sistema operativo de software libre basado en *Debian* y diseñado para su utilización en *Raspberry Pi.* El núcleo de *Debian* es Linux, y se trata de un software diseñado por voluntarios de todo el mundo con el objetivo de crear un sistema operativo totalmente libre, siempre utilizando herramientas de GNU.

Python [27]

Python es un lenguaje de programación de alto nivel y de código abierto desarrollado por la empresa *Python software Foundation*. Se trata de un lenguaje de programación orientado a diferentes objetivos, desde la programación orientada a objetos hasta la programación imperativa y funcional. Su diseño se basa en la simplicidad del lenguaje, es decir, en ser un código de fácil lectura e implementación. *Python* facilita el trabajo del programador al no ser necesaria la compilación de los códigos desarrollados, además de la gran variedad de librerías estándar que posee.

Python comenzó a concebirse a finales de los 80 y su implementación fue iniciada en diciembre de 1989 por Guido van Rossum en el CWI en Países Bajos, como un sucesor al lenguaje de programación ABC, capaz de manejar excepciones e interactuar con el sistema operativo Amoeba.

***OpenCV* [28]**

OpenCV es una biblioteca multiplataforma de carácter libre, dirigida principalmente hacia el campo de la visión artificial. Esta biblioteca está diseñada para ser empleada en distintos lenguajes de programación tales como *C, C++, Python, Java y Matlab*, teniendo un correcto funcionamiento en prácticamente todos los sistemas operativos.

Se trata de una biblioteca con más de 2500 algoritmos, cuyo objetivo es el de proporcionar un entorno de fácil uso y con una alta eficiencia. Con esta biblioteca se pueden realizar numerosas acciones tales como la identificación de objetos o caras, eliminar ojos rojos, reconocimiento de escenarios…

Las aplicaciones más destacadas en las que se emplea esta biblioteca son la detección de intrusos, la monitorización de equipamientos, el guiado de robots…

2.3.2. Fundamentos De La Visión Estereoscópica

Como ya comentamos, nuestro sistema se basa en la visión estereoscópica. Podemos definir este término como la capacidad que tiene un ser vivo de integrar las imágenes captadas por cada uno de los ojos en una sola, dicha tarea es realizada por el cerebro. Estas imágenes capturadas por cada uno de los ojos son tomadas desde dos puntos separados una distancia concreta y desde ángulos distintos (véase figura 27), esto es lo que conocemos como disparidad, y es lo que nos permite apreciar la profundidad y el relieve de los objetos.

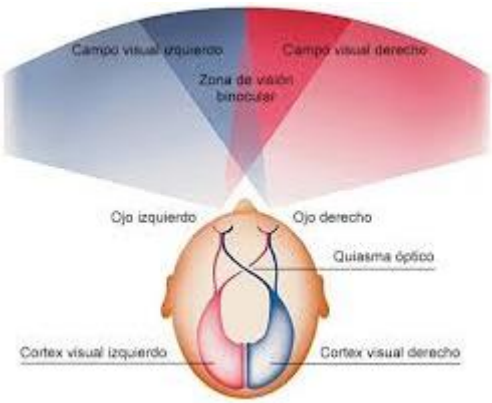


Figura 27: Visión humana [29]

Para conseguir reproducir la visión humana, se utilizan un par de cámaras con una distancia entre ellas aproximadamente igual a la existente entre los ojos de una persona, formando lo que se conoce como par estereoscópico. Posteriormente estas imágenes serán superpuestas, consiguiendo el efecto estereoscópico, tal y como se muestra en la figura 28.

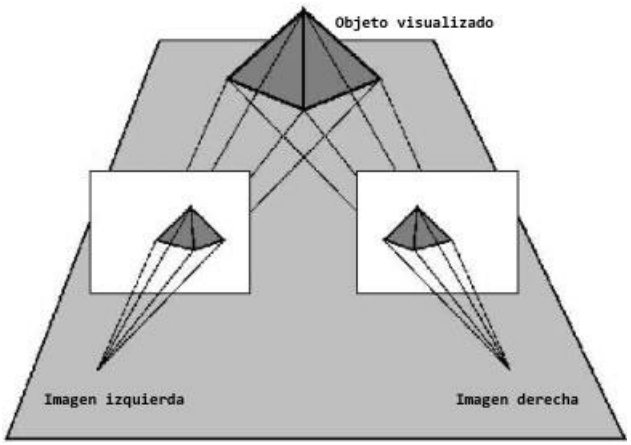


Figura 28: Visión estereoscópica [30]

Para llevar a cabo la superposición de ambas imágenes se utiliza la técnica de la correlación. Con esta técnica se buscan las similitudes existentes entre las dos imágenes que capturan ambas cámaras. Para ello se recorren dichas imágenes pixel a pixel en busca de aquellos puntos en los que el valor de correlación es máximo, encontrando en esos puntos la equivalencia. Estos puntos equivalentes nos permitirán la superposición de las dos imágenes.

La correlación tiene la siguiente expresión matemática:

$$C(i,j) = \sum_{x=0}^{L-1} \sum_{y=0}^{K-1} w(x,y) f(x+i, y+j)$$

donde w(x,y) será el valor digital del pixel con coordenadas (x,y) en la ventana de análisis de las imágenes cuyo tamaño es K x L, siendo comparado con la imagen completa f(x,y) cuyo tamaño M x N.

Podemos definir ventana como una matriz de un número determinado de píxeles, con la que se va a ir recorriendo la imagen completa para la búsqueda de las similitudes entre ambas imágenes.

De manera gráfica, la correlación puede ilustrarse como en la figura 29.

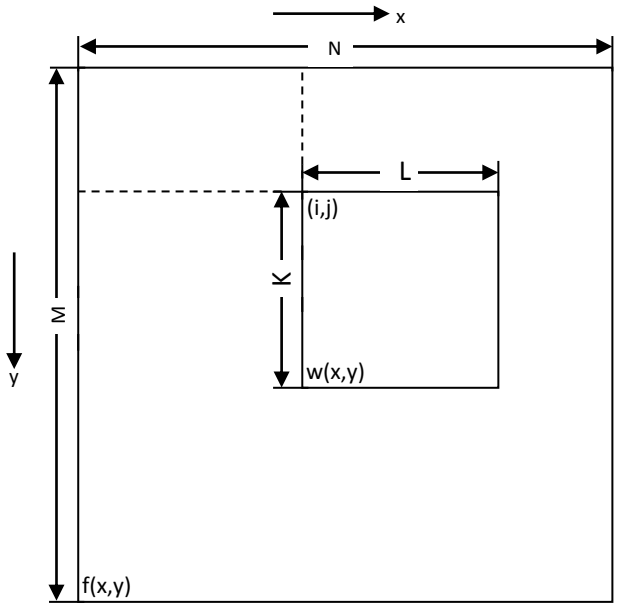


Figura 29: Procedimiento de correlación

La ventana anteriormente definida, llevará a cabo un barrido de toda la imagen para la búsqueda de la máxima similitud o correlación entre ambas imágenes, siendo el máximo valor de la expresión C(i,j) el que indica que los pixeles con los que se ha calculado dicho valor son los pixeles coincidentes entre ambas.

En el caso del problema que nos atañe, el de extraer el mapa de disparidad de las imágenes capturadas por las cámaras, se pueden utilizar dos tipos de correlación, la correlación basada en el área o la correlación basada en características.

- Correlación basada en área: a partir de un pixel de una imagen, se sigue la línea epipolar que pasa por él y realizar una búsqueda en la otra imagen para encontrar el pixel con la intensidad más parecida a nuestro pixel.

- Correlación basada en características: esta correlación utiliza estructuras de nivel superior al pixel, para establecer la correspondencia entre ambas imágenes a partir de las características de las mismas, como pueden ser los bordes de objetos o regiones.

2.4. Sistema De Salida

2.4.1. Gafas Y Cascos

Como ya sabemos, nuestra aplicación se basa en la captura y tratamiento de imágenes, hasta conseguir unos contornos con diferentes colores en función de la distancia con objeto de ayudar a personas con baja visión. Debido a esto, es necesario un sistema de salida donde podamos mostrar estas imágenes capturadas y tratadas, donde el usuario final podrá recibir dichas imágenes como apoyo a su visión en su vida diaria.

Dado que el sistema debe ser completamente portable, debemos realizar una búsqueda de las diversas tecnologías existentes, ya que no se puede entregar a una persona un sistema que sea muy pesado, así como no se le puede entregar un producto cuyo rendimiento sea muy bajo, sino que debemos buscar un sistema cuyo diseño sea apropiado para utilizar en las actividades realizadas en la vida cotidiana.

Actualmente, este es un campo en constante desarrollo. Para nuestra aplicación podemos destacar productos con dos tipos de tecnología, los que utilizan micropantallas o los que utilizan microproyectores. La principal diferencia entre ellos, es que el primero muestra la imagen a través de una pantalla con retroiluminacion dirigida directamente al ojo, mientras que los segundos proyectan la imagen sobre un panel que refleja las imágenes enviándolas hasta nuestro ojo. Cualquiera de los dos sistemas podría ser utilizado en nuestra aplicación, sin embargo, el caso de los microproyectores resulta un poco más complejo dada la necesidad de un proyector más una pantalla donde reflejar la imagen. Por su parte, las micropantallas irían directamente acopladas a la gafa dado una mayor sensación de normalidad al usuario.

La decisión de elegir un sistema con micropantallas o con microproyectores se puede fundamentar en el deseo o necesidad de mantener la poca o mucha visión de la que el paciente disponga. Ya que podríamos elegir una tecnología “see-through” que permite ver a través de la pantalla, o elegir una tecnología completamente opaca en la que la única visión disponible sea la aportada por las pantallas. En este caso, si se eligiera un sistema “see-through” sería más fácil utilizar un sistema basado en microproyectores, reflejando la imagen sobre el cristal de la gafa. Mientras que si queremos que toda la visión del paciente se centre en la imagen de la pantalla, sería más cómodo y con mejor calidad un sistema basado en micropantallas, cuyo efecto visto desde el exterior podría ser parecido a unas gafas de sol.

También hay que tener en cuenta si se quiere utilizar un sistema con dos pantallas o con una sola. Si se desea utilizar un sistema con dos pantallas, la estructura con dos proyectores tendría una envergadura mayor que si se utilizan dos micropantallas.

A continuación se muestran algunos dispositivos con estas tecnologías y sus características más importantes:

M3 de Virtual Realities

- Monocular
- Display: 800 x 600
- Conexión: DVI-D a PC
- Peso: 100 gramos
- Campo de visión: 32 grados
- Tipo de visión: see-through
- Precio: 2.700 €

Figura 30: M3 Virtual Realities

- http://trivisio.en.forbuyers.com/product/8970453 (9/06/2015)

Addvisor150 de Virtual Realities

- Binocular
- Display: 1280 x 720
- Conexión: wireless o HDMI
- Peso: 1,83 kilogramos
- Campo de visión: 45 grados
- Tipo de visión: opaco
- Precio: 1.299 €

Figura 31: Sony HMZ-T3W Head Mounted 3D

- http://www.sony.co.uk/electronics/head-mounted-display-products/hmz-t3w (9/06/2015)

SV-6 PC Viewer de MicroOptical

- Binocular
- Display: 1280 x 720
- Conexión:
- Peso: 1,87 kilogramos
- Campo de visión: 45 grados
- Tipo de visión: opaca
- Precio: 999 €

Figura 32: Sony HMZ-T2 Head Mounted 3D

- http://www.sony.co.uk/electronics/head-mounted-display-products/hmz-t2#product_details_default (9/06/2015)

SXGA61 3D de Trivisio

- Binocular
- Display: 640 x 480
- Conexión: USB2.0
- Peso: 699 gramos
- Campo de visión: 35 grados
- Tipo de visión: opaca
- Precio: 820 €
- http://www.amazon.co.uk/MaxSight-Eyewear-Multimedia-iTheater-Playstation3-x/dp/B005IE5V34 (9/06/2015)

Figura 33: MaxSight 3D Viewer

Wrap 1200DXAR de Vuzix

- Binocular
- Display: 852 x 480
- Conexión: VGA
- Peso: menos de 1 kg
- Campo de visión: 35 grados
- Tipo de visión: see-through
- 2 cámaras incorporadas
- Precio: 1.499 $
- **http://www.vuzix.com/augmented-reality/products_wrap1200dxar/** (9/06/2015)

Figura 34: Wrap 1200DXAR Vuzix

STAR 1200XLD de Vuzix

- Binocular
- Display: 852 x 480
- Conexión: VGA
- Peso: menos de 1 kg
- Campo de visión: 35 grados
- Tipo de visión: see-through
- 1 cámara incorporada
- Precio: 4.999 $
- **http://www.vuzix.com/augmented-reality/products_star1200xld/** (9/06/2015)

Figura 35: STAR 1200XLD Vuzix

Para nuestra aplicación, no resulta estrictamente necesario tener un sistema de salida de tipo “see-through” u opaco dado que para mostrar las profundidades utilizaremos contornos de diversos colores. Tras habernos puesto en contacto con el IOBA (Instituto de OftalmoBiología Aplicada) y saber su opinión sobre el tema, vamos a escoger un sistema que permita ver a través de las lentes, para no eliminar el resto visual de posibles pacientes cuyo ángulo de visión sea superior al de los displays utilizados.

Para el desarrollo del proyecto se han utilizado las gafas STAR 1200XLD de Vuzix, ya que son de las que se disponía en el grupo (GDAF), añadiéndole la facilidad de tener incorporado a las gafas el sistema de adquisición de imágenes, es decir, la cámara *Minoru 3D WebCam* tal y como se muestra en la figura 36.

De cualquier modo, los algoritmos desarrollados en este proyecto pueden ser utilizados en otros dispositivos de salida de menor coste, en caso de que estos pudieran ser adquiridos, permitiendo su evaluación en un proyecto de investigación.

Figura 36: Sistema de salida

2.5. Montaje Final Del Sistema

El sistema completo se puede concebir en un diagrama de bloques tan sencillo como el que se muestra en la figura 37:

Figura 37: Esquema de los bloques del sistema

El sistema de salida y el sistema de adquisición de imágenes, como ya comentamos, fueron obtenidos en el laboratorio con el montaje ya realizado. Por el contrario, el sistema de procesamiento utilizado, la *RaspberryPi 2 Model B,* fue comprada y venía tan solo incluida la placa, sin ningún tipo de protector, por lo que se ha impreso una carcasa con una impresora 3D perteneciente al GDAF-UC3M, cuyo diseño ha sido realizado en el programa de diseño gráfico *Sketchup* (véase figura 38).

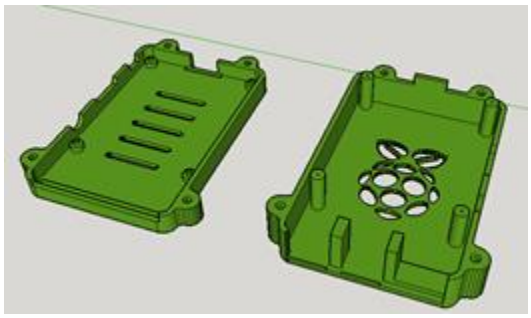


Figura 38: Diseño carcasa RaspberryPi

La impresora utiliza ha sido la disponible en el laboratorio, marca *Makerbot* modelo *Replicator G* (figura 39). A partir del modelo implementado en *Sketchup*, se genera un archivo .stl. Dicha extensión es la que es capaz de interpretar el programa de la impresión 3D. Ya solo queda generar el "gcode", este código traduce el diseño gráfico de la pieza a fabricar a un lenguaje de programación entendible por la interfaz de la impresora. Por último hay que lanzar este "gcode" en la impresora y esperar que la pieza esté lista.

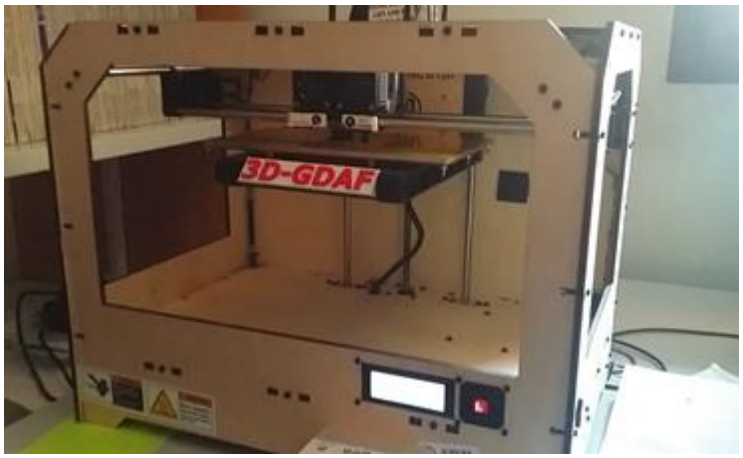


Figura 39: impresora MakerBot Replicator G

Finalmente, el diseño del sistema de procesamiento queda como se muestra en la figura 40.

Figura 40: diseño final sistema de procesamiento

En la figura 42 podemos ver la envergadura del sistema completo, comprobando que se trata de un sistema totalmente portable y que se puede utilizar en cualquier lugar, como queda demostrado en la figura 41.

Figura 41: Autor de este TFG utilizando el sistema

Figura 42: Sistema completo

Capítulo 3: Pruebas Iniciales Y Resultados Experimentales

3.1. Diseño De Los Algoritmos

El proyecto requiere la implementación de diversos algoritmos desarrollados, como se comentó anteriormente en *Python* con ayuda de las librerías de *OpenCV*.

El código implementado consta de 5 programas diferentes, cuya relación es directa: Captura.py, Calibracion.py, Mostrar.py, Mapa.py, Tecla.py.

El primer programa sirve para la captura de las imágenes para la calibración, el segundo programa realiza la calibración, el tercer programa muestra la calibración realizada y el cuarto programa calcula el mapa de disparidad y crea los contornos asociados a dicho mapa de disparidad. Por último tenemos el quinto programa asociado al último programa mencionado, que permite la modificación del valor de ciertos parámetros para conseguir un mapa de disparidad óptimo. A continuación, en la figura 43 se muestra el proceso que sigue nuestro sistema:

Figura 43: proceso del sistema

Con los algoritmos desarrollados se trata de calibrar, mediante Software, las cámaras utilizadas para posteriormente ser capaces de extraer un mapa de disparidad sin errores.

Captura.py (ver anexo A.2.1)

Se encarga de analizar cada *frame* que se captura desde cada una de las cámaras buscando un patrón con forma de tablero de ajedrez, que posteriormente será utilizado para la calibración de las cámaras de nuestro par estéreo. Para ello sigue el proceso mostrado en la figura 44.

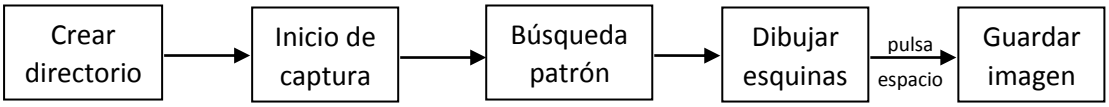


Figura 44: Proceso Captura.py

Tras crear el directorio y comenzar la captura mediante la función de OpenCV "QueryFrame, se utiliza la función "FindChessboardCorners"[2] para la localización del patrón del tablero de ajedrez. Una vez localizado este patrón, el programa dibuja sobre cada *frame* una serie de líneas que unen las esquinas del tablero de ajedrez (ver figura 45), utilizando la función "DrawChessboardCorners"[3]. Además, mediante el pulsado de la "barra espaciadora", podremos ir guardando las imágenes capturadas por las cámaras en las cuales ha sido identificado el patrón del tablero. Estas imágenes serán almacenadas en un directorio que el mismo programa te permite crear nada más arrancarlo.

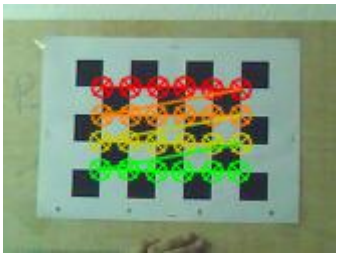


Figura 45: Patrón ajedrez encontrado

Calibracion.py (ver anexo A.2.2)

El código implementado nos permite realizar un análisis de las imágenes capturadas en el primer programa, a partir de las cuales realiza una calibración de las cámaras para su correcto alineamiento, para lo que se sigue el proceso mostrado en la figura 46.

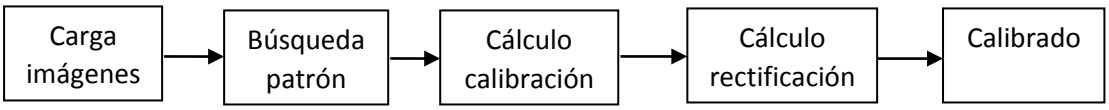


Figura 46: Proceso Calibracion.py

En primer lugar este programa carga las imágenes del primer programa del directorio en el que han sido guardadas y vuelve a realizar una búsqueda de las esquinas del tablero de ajedrez, una vez encontradas dichas esquinas, se utiliza la función "FindCornerSubPix", que nos permite encontrar dichas esquinas a nivel de sub-pixeles para poder hacer una calibración más precisa.

---

[2] FindChessboardCorners: la función identifica el patrón de un tablero de ajedrez definido por las esquinas interiores del mismo, las cuales utiliza para identificar el tablero de ajedrez. Esta función devuelve un 0 si no detecta el patrón del ajedrez, y retorna un 1 si el patrón es identificado.

[3] DrawChessboardCorners: esta función está ligada a la anterior, ya que se encarga de remarcar y unir las esquinas identificadas por la misma.

Posteriormente se emplea la función “StereoCalibrate” mostrada en la figura 47, esta función se encarga de realizar las transformaciones necesarias para formar un par estéreo. Esta función se basa en el hecho de que, conociendo la posición relativa de una cámara respecto de la otra, dada la imagen de una cámara (R1, T1) es posible conocer la posición de los objetos de la imagen capturada por la otra cámara (R2, T2).

$$R2 = R * R1T2 = R * T1 + T$$

Esta función lleva a cabo un proceso iterativo para conseguir la calibración óptima. Este proceso termina, en este caso, cuando el número de iteraciones ha llegado a 30 ó cuando se ha conseguido una calibración cuyo error es inferior a 1e-6. Se han elegido estos valores tras realizar varias pruebas experimentales, en las que se ha llegado a la conclusión que permitir un número mayor de iteraciones requiere mucho más cómputo y el resultado no es mucho mejor. En cuanto al valor del error, se ha escogido ese valor por ser el valor defecto que aporta la función y comprobar que es el valor más utilizado para dicha función.

```
cv.StereoCalibrate(object_Points, image_PointsL, image_PointsR, point_counts,
                   leftIntrinsics, leftDistortion, rightIntrinsics, rightDistortion,
                   (imageWidth, imageHeight), R, T, E, F,
                   (cv.CV_TERMCRIT_ITER+cv.CV_TERMCRIT_EPS, 30, 1e-6), cv.CV_CALIB_FIX_PRINCIPAL_POINT )
```

Figura 47: Función de calibración para visión estereoscópica

Tras realizar dicha calibración, se utiliza la función “StereoRectify”, que calcula las matrices de desplazamiento para que los planos de captura de ambas cámaras estén en el mismo plano. Para ello, utiliza los datos obtenidos de la función anterior y genera dos matrices de desplazamiento.

En el caso de la rectificación horizontal nos encontramos con las siguientes matrices:

$$P1 = \begin{bmatrix} f & 0 & cx_1 & 0 \\ 0 & f & cy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$P2 = \begin{bmatrix} f & 0 & cx_2 & T_x * f \\ 0 & f & cy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

En caso de la rectificación vertical tenemos las siguientes matrices:

$$P1 = \begin{bmatrix} f & 0 & cx & 0 \\ 0 & f & cy_1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$P2 = \begin{bmatrix} f & 0 & cx & 0 \\ 0 & f & cy_2 & T_y * f \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Siendo:

P1, P2- matrices de desplazamiento

CX, CY- coordenadas línea epipolar

T_X - desplazamiento horizontal entre las cámaras.

T_Y- desplazamiento vertical entre las cámaras.

Las tres primeras columnas de P1 y P2 serán las nuevas matrices de calibración de nuestro par estéreo. Estos valores se guardan y se pasarán a la función “InitUndistortRectifyMap”, que será la encargada de realizar la calibración de ambas cámaras. Esta función construye los mapas para el algoritmo de mapeo inverso que utiliza la función remap() para realizar la transformación correspondiente. Es decir, para cada pixel en la imagen destino, la función calcula las coordenadas correspondientes en la imagen origen.

Mostrar.py (ver anexo A.2.3)

Este código carga los archivos guardados en el programa anterior y vuelve a aplicar estos datos para realizar la calibración, mostrándola posteriormente por pantalla. De este modo puede aceptar dicha calibración, o desecharla y volver a realizar todo el proceso partiendo del primer programa de captura de imágenes.

Mapa.py (ver anexo A.2.4)

Por último se ha creado el código, que realiza el cálculo del mapa de disparidad (véase figura 48), el cual nos dará la información relativa a la distancia y la profundidad de la escena capturada.

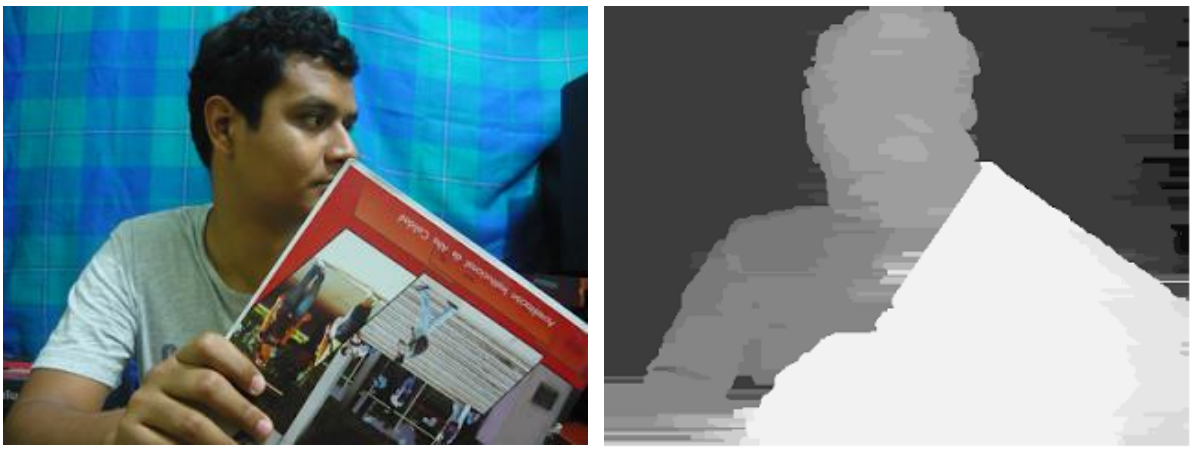


Figura 48: Mapa de disparidad [31]

Para la obtención del mapa de disparidad, en primer lugar el programa carga los archivos con los datos de rectificación guardados anteriormente y realiza nuevamente la calibración mediante la función “InitUndistortRectifyMap” ya explicada. Esto se debe a que los parámetros de calibración han sido guardados, pero el sistema de adquisición no es capaz de guardar dicha calibración, por lo que cada vez que se vaya a utilizar el sistema estéreo, es necesario realizar la calibración con los parámetros calculados anteriormente.

Posteriormente extrae del constructor de la función “StereoBM” las variables que se van a poder modificar para el ajuste del mapa de disparidad, las cuales son:

- SADWindowSize: tamaño promedio de la ventana utilizada para la correlación. Toma valores entre 5 y 255.
- preFilterType: filtro aplicado a la imagen antes de calcular la disparidad para facilitar dicho cálculo. Puede valer 0 o 1, correspondiendo con un filtro *sobel* o con una normalización de la imagen.
- preFilterSize: tamaño del pre-filtro utilizado. Toma valores entre 5 y 255.
- preFilterCap: valor de separación entre los pixeles de la imagen pre-filtrada. Toma valores entre 1 y 63.
- minDisparity: mínimo valor posible de la disparidad. Suele ser 0 pero algunos algoritmos requieren una variación de este parámetro.
- numberOfDisparities: máximo valor posible de la disparidad. Siempre es mayor que 0.
- textureThreshold: valor umbral de textura de la imagen capturada. Toma valores entre 0 y 255.
- uniquenessRatio: utilizada para encontrar la mejor similitud entre las dos imágenes. Toma valores entre 0 y 255, aunque suele ser suficiente entre 5 y 15.
- speckleRange: máxima variación de la disparidad aceptada entre los componentes u objetos conectados. Toma valores entre 0 y 255.
- speckleWindowSize: máximo tamaño de las regiones de disparidad para considerar el ruido o distorsiones presentes debido al sistema de adquisición de imágenes. Toma valores entre 0 y 255.

Posteriormente se procederá al cálculo de los contornos de nuestra captura, para lo que utilizamos la función “Canny”. Esto se debe a que a partir de proyectos anteriores [21], se ha observado que para optimizar el sistema es necesario realizar un cálculo de contornos previo a nuestro cálculo del mapa de disparidad.

Después el programa va a proceder al cálculo del mapa de disparidad, para lo cual utiliza la función “FindStereoCorrespondenceBM”. Esta función se encarga del cálculo del mapa de disparidad para nuestro par estéreo, para lo que nuestro sistema ha de estar calibrado.

Como vemos en la figura 50 del mapa de disparidad, hay varios niveles de gris: esto se debe a que cada uno de estos niveles corresponde a una distancia diferente de cada objeto al par estereoscópico. Los objetos más cercanos corresponden con el color blanco, mientras que a los objetos más lejanos les corresponde el negro, siendo las distancias intermedias los distintos niveles de gris, es decir, a mayor distancia la tonalidad del color será más oscura.

Observando la figura 49, vemos que además del cálculo de este mapa de disparidad, el programa nos permite ver la captura de cada cámara de manera independiente, así como los contornos de los objetos que hay en escena en tres colores diferentes en función de la distancia de cada uno de los objetos al par estéreo, siendo rojo el correspondiente a los

objetos más cercanos, verde para los objetos situados a media distancia y azul para aquellos objetos que se encuentran más alejados del primer plano.

Para obtener dichos contornos se han utilizado dos funciones. En primer lugar es necesaria la detección de dichos contornos, lo que se realiza mediante la función “FindContours”, esta función requiere la conversión previa de la imagen a una imagen binaria. La segunda función requerida es “DrawContours”, que se encarga de dibujar los contornos detectados por la función anterior. Adicionalmente, esta función nos permite modificar tanto el color de los contornos a mostrar como su grosor mediante la variación de algunos de sus parámetros.

Figura 50: Captura mapa disparidad

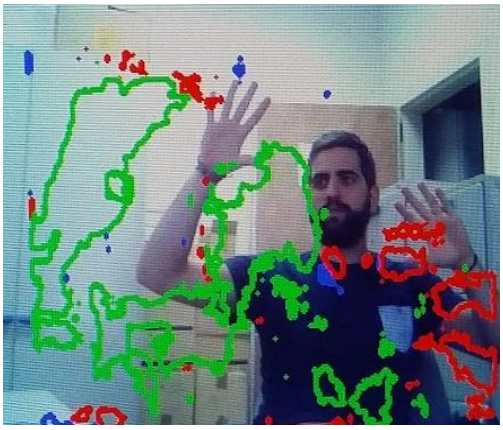


Figura 49: Captura contornos colores

Tecla.py (ver anexo A.2)

El código implementado en este programa está directamente ligado al programa Mapa.py, permitiendo la modificación de los parámetros mencionados anteriormente mediante la actuación sobre diferentes teclas del teclado aquí definidas, hasta conseguir un mapa de disparidad que nos permita identificar la escena perfectamente, teniendo cada objeto su color correspondiente en función de su distancia a las cámaras. Los parámetros más importantes para definir el mapa de disparidad son:

- SADWindowSize
- preFilterType
- preFilterSize
- preFilterCap
- minDisparity.
- numberOfDisparities
- textureThreshold
- uniquenessRatio
- speckleRange
- speckleWindowSize

Esta última ventana donde se muestran los contornos con los diferentes colores indicando la distancia, ha de ser adaptada a la vista del paciente ya que cada paciente tienes unas necesidades y un resto residual.

Para poder adaptar la imagen procesada a la visión del paciente se ha desarrollado un programa para realizar un estudio al paciente que nos permita decidir cuál es la región de la pantalla más óptima en la que colocar nuestra imagen. Además de todo esto el programa permitirá aumentar o disminuir el tamaño de la captura para poder ajustarlo a la visión del usuario.

Todo el desarrollo de este ajuste a la visión del paciente consta en el Trabajo Fin de Grado de mi compañero Jonathan Pajares Redondo.

3.2. Resultados Experimentales Del Sistema De Adquisición De Imágenes

Tras, como ya hemos comentado, haber elegido el sistema de adquisición de imágenes Minoru 3D WebCam, vamos a realizar la medida correspondiente a una captura simple realizada mediante la función de *OpenCV* "QueryFrame", con lo que podremos saber la velocidad real de captura de la cámara (que no la de los algoritmos desarrollados) frente a la velocidad que nos indica el fabricante. Hay que tener en cuenta que la velocidad de captura no solo depende de la cámara en cuestión, sino que también depende del sistema de procesamiento, principalmente de la tarjeta gráfica que éste tenga.

Este tiempo será comprobado tanto en un PC portátil convencional (marca Acer modelo Aspire 5634WLMi), como en los dos sistemas de procesamiento elegidos, es decir, RaspberryPi B y RaspberryPi 2 B.

Dado que nos encontramos ante una captura realizada a partir de dos cámaras, los valores medidos nunca van a ser fijos, ya que estos valores dependen de la iluminación de la escena, el brillo, la resolución de captura…, pero sí que van a tener un rango con el que vamos a saber más o menos el tiempo de adquisición de la imagen.

PC portátil

El tiempo empleado por el sistema para realizar la captura de la imagen, por las dos cámaras simultáneamente, toma valores entre 25 y 40 milisegundos tal y como se muestra en la figura 51.

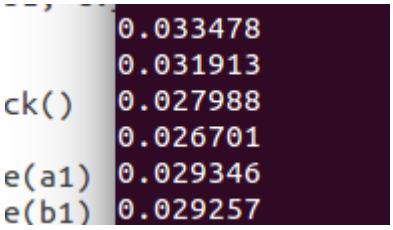


Figura 51: Tiempo de adquisición de imagen con PC

Si calculamos la inversa de la medida realizada, podemos comprobar que nuestro sistema de adquisición de imágenes tiene una velocidad de captura entre 25 y 40 fps. Si lo comparamos con la velocidad de 30fps a la que se muestra una película para evitar que se vean saltos entre frames, podemos confirmar que esta velocidad de adquisición es válida para nuestro sistema.

RaspberryPi Model B

El tiempo de captura medido en uno de los sistemas de procesamiento elegidos, RaspberryPi B, se han obtenido unos tiempos variables entre 40 y 50 milisegundos, tal como se muestra en la figura 52.

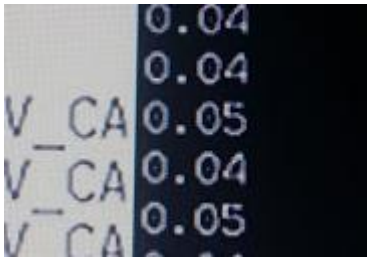


Figura 52: Tiempo de adquisición de imagen RaspberryPi B

Haciendo la inversa de los valores obtenidos, sacamos una velocidad de captura de imágenes entre 20 y 25 fps. Este valor está por debajo del valor al que podemos ver un video de modo continuo, pero su valor no es muy lejano, aunque sí que se apreciarán pequeños saltos entre un frame y el siguiente.

RaspberryPi 2 Model B

En caso de realizar la captura en el último modelo del sistema de adquisición elegido, los tiempos medidos son de 40 milisegundos, como se muestra en la figura 53.

Figura 53: Tiempo de adquisición de imagen RaspberryPi 2 B

Con esto obtendremos un valor de la velocidad de captura de 25 fps, con el que podemos considerar que la sucesión de frames puede verse de modo continuo, aunque sigue siendo una captura más lenta que la realizada en el PC.

Comparando los resultados anteriores, podemos ver que el tiempo de adquisición de la cámara a través del PC, es menor que cuando se realiza la captura a través de los otros sistemas de procesamiento. Como hemos comentado anteriormente, la tarjeta gráfica y el procesador influían en dichos tiempos, lo que podemos corroborar mediante estas medidas.

Como primera conclusión, podemos decir que nuestros dispositivos de procesamiento no son óptimos para la herramienta diseñada, dado que necesitamos un sistema que permita ver la imagen en tiempo real.

3.3. Resultados Experimentales Del Tiempo De Ejecución De Los Algoritmos

Al igual que en el apartado anterior, se van a medir los tiempos de ejecución de los programas finales en los tres dispositivos de los que se dispone, el PC mencionado anteriormente y los dos modelos de RaspberryPi.

Como ya dijimos anteriormente, nuestro proyecto está constituido por cinco programas diferentes.

1. Captura.py
2. Calibracion.py
3. Mostrar.py
4. Mapa.py
5. Tecla.py

Los 3 primeros programas no es necesario lanzarlos en el sistema de procesamiento a utilizar, ya que tan solo nos sirven para obtener los parámetros de calibración y no requieren de la característica de tiempo real. Estos parámetros se guardan en varios archivos con extensión .xml, que habrá que pasar al sistema de procesamiento junto con los programas 4 y 5.

Además, para el tercer programa, al encargarse tan solo de mostrar la calibración realizada, tampoco vamos a hacer una medición de tiempos.

Por esto, los programas 1 y 2 tan solo serán lanzados y, por tanto, medidos sus tiempos de ejecución en el ordenador, mientras que el programa 4 será lanzado tanto en el ordenador como en los dos modelos de RaspberryPi. El quinto programa no es necesario lanzarlo, simplemente debe acompañar siempre al programa 4.

MEDIDAS REALIZADAS EN EL PC PORTÁTIL

El programa correspondiente a la captura de las imágenes para la posterior calibración, toma un tiempo de ejecución con valores comprendidos entre los 300 y los 400 milisegundos por ciclo.

El segundo programa, encargado de la calibración de las dos cámaras, requiere 7.32 segundos para realizar dicha calibración.

El último programa encargado de la extracción del mapa de disparidad, toma un tiempo de ciclo de 400 milisegundos para realizar el primer ciclo completo en el que se realizan los cálculos necesarios, quedando posteriormente un tiempo de refresco para la captura de frames de entre 50 y 55 milisegundos.

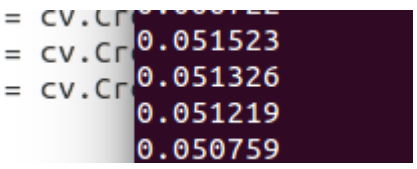


Figura 54: Tiempo de procesamiento PC

Con este dato podemos concluir que nuestro programa funciona a una velocidad de unos 18 fps (véase figura 54), quedando este valor alejado del necesario para poder percibir la imagen a tiempo real perfectamente.

MEDIDAS REALIZADAS EN *RASPBERRYPI MODEL B*

El primer ciclo del programa que calcula y muestra el mapa de disparidad conlleva 1.11 segundos de ejecución. Una vez realizada toda la configuración, el programa tiene un refresco que varía entre los 750 y los 820 ms por ciclo de refresco como podemos ver en la figura 55.

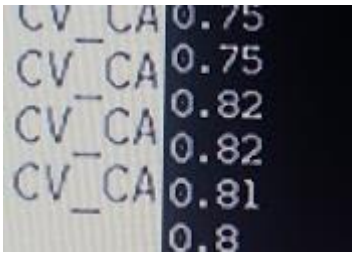


Figura 55: Tiempo de procesamiento RaspberryPi B

Calculando la inversa de estos valores, obtenemos una velocidad de captura de entre 1.21 y 1.33 fps. Este valor se encuentra muy alejado del valor buscado para conseguir una visión en tiempo real, que supondría una velocidad de captura entre 25 y 30 fps.

MEDIDAS REALIZADAS EN *RASPBERRYPI 2 MODEL B*

En el modelo RaspberryPi 2 B, el primer ciclo requiere de un tiempo de procesamiento de 780 milisegundos para realizar todos los cálculos necesarios, quedando posteriormente un tiempo de refresco de la captura variable entre 480 y 510 milisegundos, tal y como se muestra en la figura 56.

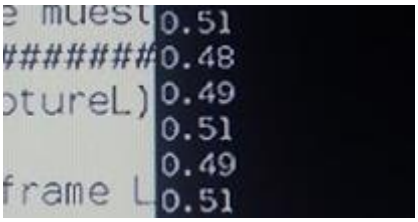


Figura 56: Tiempo de procesamiento RaspberryPi 2 B

Con estos valores obtenemos una velocidad de captura de imagen de entre 1.96 y 2.08 fps. De nuevo podemos observar que este valor se encuentra muy alejado de la velocidad de captura necesaria para poder ver el video a tiempo real.

Tras las medidas realizadas, podemos comprobar que el tiempo de ejecución por ciclo que el programa requiere, es mucho menor en el PC que en los otros dos sistemas de procesamiento. Como ya comentamos anteriormente, esto era de esperar debido a la diferencia entre los procesadores y las tarjetas gráficas de unos y otros dispositivos.

Podemos comprobar que el modelo nuevo de RaspberryPi, mejora bastante los tiempos, reduciéndolos casi hasta la mitad, pero aun así, sigue resultando ser un tiempo de procesamiento excesivamente elevado para la aplicación que se está desarrollando.

Se han realizado medidas con el PC para poder realizar una comparación de los tiempos de ejecución con un sistema de procesamiento más potente de lo que es una RaspberryPi, sin embargo el PC no puede ser una solución a nuestro sistema debido a su gran envergadura, por lo que nosotros estamos interesados en la optimización de los sistemas de procesamiento portátil y de bajo coste, como es el caso de la RaspberryPi.

Con estos resultados, se hace más firme la conclusión sacada en el apartado anterior, de que nuestro sistema de procesamiento no es válido para la aplicación deseada.

Capítulo 4: Conclusiones Y Posibles Líneas Futuras

4.1. Presupuesto

En la siguiente tabla se muestra el coste del sistema completo (ambos TFGs) dividido en capítulos, resaltando los elementos utilizados en este proyecto. Como podemos comprobar, la mayor parte del presupuesto corresponde al sistema de salida. Como ya comentamos anteiormente, este coste podría ser reducido mediante la adquisición de nuevos sistemas de salida más económicos en futuros proyectos.

CÓDIGO	UNIDAD DE MEDIDA	DESCRIPCIÓN	CANTIDAD	PRECIO (UNITARIO)	IMPORTE TOTAL
		CAPÍTULO 1: Sistema de adquisición			
1.01	unidades	Minoru Web-Camera 3D	1	46,09 €	46,09 €
			TOTAL CAPÍTULO 1: Sistema de adquisición		46,09 €
		CAPÍTULO 2: Sistema de procesamiento			
02.01	unidades	Bobina de plástico ABS Filamento de plástico ABS negro de 1.75mm. 1 Kg	1	17,95 €	17,95 €
02.02	unidades	Interruptores Interruptor basculante de 2 posiciones negro	10	0,83 €	8,30 €
02.03	unidades	Placa de puntos. Placa C.I. 2 cuadrados 100x160MM	1	7,00 €	7,00 €
02.04	unidades	Tarjeta microSDHC Verbatim 16GB Clase 10	1	15,00 €	15,00 €
02.05	unidades	Bateria Power Bank A5 Classic 2600mAh	1	5,99 €	5,99 €
02.06	unidades	Raspberry Pi 2 Modelo B	1	35,30 €	35,30 €
			TOTAL CAPÍTULO 2: Sistema de procesamiento		90 €
		CAPÍTULO 3: Sistema de salida			
03.01	unidades	Gafas Vuzix STAR 1200XLD	1	4.999 €	4.999 €
03.02	unidades	Cable adaptador, VGA hembra HDMI macho	1	10,75 €	10,75 €
			TOTAL CAPÍTULO 4: Sistema de salida		5.010 €
		CAPÍTULO 4: Honorarios Ingenieros			
05.01	horas	Honorario a los dos ingenieros al cargo del proyecto, suponiendo un salario para cada uno de 26 €/hora	240	26 €	6.240 €
			TOTAL CAPÍTULO 4: Honorarios Ingenieros		6.240 €
				TOTAL	11.386 €

4.2. Conclusiones

Llegando al final de este proyecto, podemos concluir que el resultado no ha sido el esperado al comienzo del mismo.

En primer lugar, comentar que la estructura de la cámara utilizada nos ha provocado ciertas dificultades con la obtención del mapa de disparidad, ya que los ejes horizontales de ambas cámaras no están perfectamente alineadas provocando que, tras realizar la calibración, el mapa de disparidad obtenido no es óptimo teniendo errores con la detección de las profundidades. Sin embargo tras ciertas pruebas y modificaciones, se ha conseguido eliminar este error modificando ciertos parámetros del programa. El principal valor que mejora sustancialmente dicha calibración es uno de los parámetros de la función "StereoCalibrate" que nos permita realizar dicha calibración fijando el punto principal de las imágenes.

Finalmente se ha podido comprobar que el sistema implementado funciona de modo correcto, realizando la calibración de las cámaras y obteniendo el mapa de disparidad, además de los contornos de la escena capturada con los distintos colores en función de la distancia de los objetos.

El objetivo principal del proyecto era el de comprobar si, con los sistemas propuesto para el desarrollo de esta herramienta, era posible conseguir una aplicación con funcionamiento a tiempo real. Sin embargo, hemos podido comprobar que la velocidad de procesamiento del sistema no es adecuada para la aplicación deseada, por lo que nuestro sistema queda descartado, convirtiéndose en necesaria la búsqueda de sistemas alternativos o posibles mejoras de nuestro sistema de procesamiento, algunas de las cuales serán propuestas en líneas futuas.

También se ha obtenido un resultado satisfactorio en cuanto a la necesidad de realización de un sistema de bajo coste con un presupuesto de 5150 €. Hay que tener en cuenta que más de un 95% de este presupuesto se corresponde con nuestro sistema de salida siendo posible, como ya hemos comentado con anterioridad, la disminución de esta unidad de presupuesto mediante la adquisición de nuevos sistemas de salida que conlleven un menos coste. Sin tener en cuenta el sistema de salida, podemos ver que nuestro sistema ha costado unos 150€ más honorarios, pudiendo corroborar que el objetivo del bajo coste ha sido cumplido.

Desde el punto de vista personal, se puede afirmar que el objetivo ha sido cumplido, ya que ha sido posible afianzar conocimientos adquiridos a lo largo de la carrera y permitiendo emplear dichos conocimientos en una aplicación totalmente real. Además de considerar los conocimientos adquiridos como conocimientos de gran interés ya que incluyen ámbitos muy comunes en la vida de un ingeniero como es la programación, y otros muy atractivos a nivel personal como es el tratamiento de imagen.

Además y gracias a algunas de las visitas realizadas a lo largo del proyecto, la satisfacción personal ha sido de las más importantes a lo largo de mi vida, tras observar a personas con discapacidad emocionarse gracias a nuestro sistema.

4.3. Líneas Futuras

Una de las necesidades tras la finalización de este proyecto, consiste en construir un sistema estereoscópico perfectamente alineado con dos cámaras independientes. Estas cámaras han de tener las mismas características para que el sistema funcione correctamente. Con esto podemos facilitar al programa la ejecución de la calibración. El porqué de esta necesidad se basa en que la cámara utilizada no tenía perfectamente alineadas ambos objetivos, lo que dificulta la calibración del par estéreo, provocando que el resultado sea algo peor del esperado ya que la calibración no es 100% precisa.

Adicionalmente y antes de construir dicho sistema, se deben realizar pruebas con otras cámaras dobles para comprobar si el problema observado era un problema particular de nuestra cámara u ocurre en diferentes modelos.

Una necesidad fundamental, es la de tratar de implementar nuestra aplicación en un sistema de procesamiento distinto al utilizado ya, que como hemos podido comprobar, éste resulta ser el cuello de botella de nuestro sistema. Si bien, es posible mejorar el sistema de procesamiento utilizado de distintas maneras, como podría ser mediante el paralelizado de ciertos procesos con la GPU (Graphics Processing Unit), o la aceleración por hardware.

Con el paralelizado de procesos con la GPU, se trataría de paralelizar los procesos entre la tarjeta gráfica y el procesador de nuestro dispositivo. De este modo, por ejemplo, podría realizarse la captura de la imagen en el procesador del dispositivo, mientras que la extracción del mapa de disparidad y la generación de los bordes con sus respectivos colores podría realizar simultáneamente por medio de la GPU, con lo que nuestra velocidad debería ser mucho mayor.

Con aceleración hardware, la posibilidad pasa por incluir a nuestro procesador una FPGA externa que se encargue simplemente de acelerar el proceso, es decir, nuestro procesador se encargaría de la captura de las imágenes y todo esto pasaría a través de la FPGA para volver de nuevo a nuestro sistema de procesamiento para ser mostrado en las pantallas, sin embargo al ser devuelto a nuestro sistema la velocidad va a ser muy superior a la anterior debido a la FPGA añadida.

Otro de los objetivos a cumplir a posteriori de este proyecto, es el de realizar un sistema aún más portable y con una mejor estética, de modo que se pueda llevar puesto en cualquier lugar de modo completamente disimulado, sin que nadie pueda percatarse de que el usuario lleva este sistema.

La última, y quizás más importante de las necesidades, es la de que el sistema completo sea puesto a prueba por los potenciales usuarios, pudiendo conseguir el *feedback* necesario para pensar en otras posibles mejoras, ya que debido al tiempo, esto no ha sido posible realizarlo.

Bibliografía

[1] http://www.sidar.org/recur/desdi/usable/dudt.php

Última vez consultada: 15/04/2015

[2] http://www.ioba.med.uva.es/index_00.php?&op=pac.pro.baj

Última vez consultada: 23/04/2015

[3] https://www.nei.nih.gov/healthyeyes/spanish/myopia_sp

Última vez consultada: 23/04/2015

[4] http://blog.arisvision.com/2013/05/abc-de-la-miopia.html

Última vez consultada: 20/06/2015

[5] https://www.nei.nih.gov/healthyeyes/spanish/hyperopia_sp

Última vez consultada: 23/04/2015

[6] http://www.maculadt.com/afeccion/8/miopia-hipermetropia-y-astigmatismo

Última vez consultada: 20/06/2015

[7] https://www.nei.nih.gov/health/espanol/astigmatismo/astigmatismo

Última vez consultada: 23/04/2015

[8] http://www.arisvisionjuarez.com/PadAstigmatismo/PadAstigmatismo

Última vez consultada: 20/06/2015

[9] https://www.nei.nih.gov/health/espanol/amd_paciente

Última vez consultada: 23/04/2015

[10] http://blog.ulloaoptico.com/2012/11/salud-visual-lo-que-hay-que-saber-sobre-degeneracion-macular-asociada-a-la-edad-dmae/

Última vez consultada: 20/06/2015

[11] https://www.nei.nih.gov/health/espanol/glaucoma_paciente

Última vez consultada: 23/04/2015

[12] http://effectivehealthcare.ahrq.gov/index.cfm/search-for-guides-reviews-and-reports/?productid=1478&pageaction=displayproduct

Última vez consultada: 20/06/2015

[13] https://www.nei.nih.gov/health/espanol/cataratas/cataratas_paciente

Última vez consultada: 23/04/2015

[14] http://clinicasalva.es/cataratas-3/

Última vez consultada: 20/06/2015

[15] https://www.nei.nih.gov/health/espanol/retinopatia

Última vez consultada: 23/04/2015

[16] http://drplancarte.com/padecimientos/retinopatia-diabetica/

Última vez consultada: 20/06/2015

[17] http://www.imo.es/pacientes/preguntas-frecuentes/existe-algun-tratamiento-que-frene-el-deterioro-de-la-retinosis-pigmentaria/

Última vez consultada: 23/04/2015

[18] http://www.imo.es/2013/01/29/el-imo-acoge-el-congreso-internacional-de-retina-trends-in-surgical-medical-retina-2013/

Última vez consultada: 22/06/2015

[19] http://www.retinitis.cl/retinitis_pigmentosa.php

Última vez consultada: 20/06/2015

[20] http://www.baja-vision.org/rehabilitacion7.htm

Última vez consultada: 24/04/2015

[21] https://www.researchgate.net/profile/Eli_Peli/publication/228347245_Vision_multiplexing_An_optical_engineering_concept_for_low-vision_aids/links/00b495295d754900de000000.pdf

Última vez consultada: 22/06/2015

[22] PFC Carlos Barranco: *Diseño e implementación de algoritmos de tratamiento de imágenes para ayudas técnicas visuales usando HMDS.* Proyecto que trata de desarrollar una herramienta que muestre por una pantalla un mapa de distancias de la escena capturada.

[23] PFC Francisco Collado: *Sistema de ayuda a la movilidad para personas con baja visión.* Proyecto que trata de facilitar una herramienta que capture una escena y muestre por pantalla los contornos de la misma.

[24] https://es.wikipedia.org/wiki/Muestreo_temporal

Última vez consultada: 22/06/2015

[25] https://www.raspbian.org/

Última vez consultada: 8/06/2015

[26] https://www.debian.org/index.es.html

Última vez consultada: 8/06/2015

[27] http://www.alegsa.com.ar/Dic/python.php

Última vez consultada: 5/05/2015

[28] http://opencv.org/

Última vez consultada: 5/05/2015

[29] http://grlum.dpe.upc.edu/manual/fundamentosIluminacion-laVision.php

Última vez consultada: 20/06/2015

[30] http://www.teknoplof.com/2010/04/21/el-cine-en-3d-4d-y-6d/

Última vez consultada: 20/06/2015

[31] https://sites.google.com/site/vichugof/mapas-disparidad-caseras

Última vez consultada: 20/06/2015

Anexos

A.1. Enlaces A Hojas De Características

- RaspberryPi Model B

http://docs-europe.electrocomponents.com/webdocs/127d/0900766b8127da4b.pdf

- RaspberryPi 2 Model B

http://www.adafruit.com/pdfs/raspberrypi2modelb.pdf

- Minoru 3D Webcam

http://www.minoru3d.com/

- Minoru 3D Webcam

http://www.vuzix.com/augmented-reality/products_star1200xld/

A.2. Código Implementado

A.2.1. Captura.py

```
#! /usr/bin/env python
# -*- encoding: utf-8 -*-

import sys
import os
import time
import shutil
import cv2.cv as cv

captureWidth = 320
captureHeight = 240
cam1 = 0
cam2 = 1
board_w = 7     # Número de esquinas interiores en la horizontal del tablero de ajedrez
board_h = 4     # Número de esquinas interiores en la vertical del tablero de ajedrez

if __name__ == '__main__':
   ############################################################
   # Se establece comunicación con la webcam
   ############################################################
   captureL = cv.CaptureFromCAM(cam1)
   if not captureL:                                  # Control errores
      print "Error al inicializar WebCam"           # Mensaje de error
      del(captureL)                                 # Cierre conexión
      sys.exit(1)                                   # Salir aplicación
   # Tamaño de la ventana
   cv.SetCaptureProperty(captureL, cv.CV_CAP_PROP_FRAME_WIDTH, captureWidth)
   cv.SetCaptureProperty(captureL, cv.CV_CAP_PROP_FRAME_HEIGHT, captureHeight)

   captureR = cv.CaptureFromCAM(cam2)
   if not captureR:                                  # Control errores
      print "Error al inicializar WebCam"           # Mensaje de error
      del(captureR)                                 # Cierre conexión
      sys.exit(1)                                   # Salir aplicacion
   # Tamaño de la ventana
   cv.SetCaptureProperty(captureR, cv.CV_CAP_PROP_FRAME_WIDTH, captureWidth)
   cv.SetCaptureProperty(captureR, cv.CV_CAP_PROP_FRAME_HEIGHT, captureHeight)
   ############################################################
```

```
  namefolder = raw_input("Nombre de la carpeta: ")

# Comprobación del directorio
if os.path.isdir(namefolder) is True:
   while os.path.isdir(namefolder):
      tecla = raw_input("La carpeta ya existe. Desea eliminarla? (S/n) ")
      if tecla == 'S':
         print "Borrando archivos y directorio en ./"+namefolder+" ..."
         time.sleep(2)
         shutil.rmtree(namefolder, True)
         print "...borrado"
         os.mkdir(namefolder)
         print "Nueva carpeta creada ./"+namefolder
         time.sleep(2)
         break
      if tecla == 'n':
         namefolder = raw_input("Nuevo nombre de la carpeta: ")
         if os.listdir(namefolder) == False:
            os.mkdir(namefolder)
            break
else:
   os.mkdir(namefolder)

namefile = raw_input("Nombre de inicio de los ficheros: ")

# Variable indice
i = 0

print ("Pulse espacio para realizar una captura y ESC para salir")

cv.NamedWindow("Chessboard WebCam L", cv.CV_WINDOW_AUTOSIZE) # Ventana
cv.NamedWindow("Chessboard WebCam R", cv.CV_WINDOW_AUTOSIZE) # Ventana
```

```
while True:
    ####################################################
    # Se captura un frame
    ####################################################
    frameL = cv.QueryFrame(captureL)        # Captura frame
    if frameL is None:                      # Control de errores
        print "Error captura frame"         # Mensaje de error
        break                               # Salir del WHILE
    frameR = cv.QueryFrame(captureR)        # Captura frame
    if frameR is None:                      # Control de errores
        print "Error captura frame"         # Mensaje de error
        break                               # Salir del WHILE
    ####################################################

    frameChessboardL = cv.CloneImage(frameL);
    frameChessboardR = cv.CloneImage(frameR);

    # Se analiza el frame en busca del tablero de ajedrez
    (foundL, cornersL) = cv.FindChessboardCorners(frameL, (board_w, board_h))
    (foundR, cornersR) = cv.FindChessboardCorners(frameR, (board_w, board_h))

    # Si se detecta el patrón se muestra sobre la imagen
    if foundL == 1 and foundR == 1:
        cv.DrawChessboardCorners(frameChessboardL, (board_w, board_h), cornersL, foundL)
        cv.DrawChessboardCorners(frameChessboardR, (board_w, board_h), cornersR, foundR)
        cv.ShowImage('Chessboard WebCam L', frameChessboardL)
        cv.ShowImage('Chessboard WebCam R', frameChessboardR)
        # Si se pulsa el ESPACIO se toma una captura
        if tecla == 32:
            i = i+1
            cv.SaveImage("./"+namefolder+"/"+namefile+"L"+str(i).zfill(2)+".jpg", frameL)
            cv.SaveImage("./"+namefolder+"/"+namefile+"R"+str(i).zfill(2)+".jpg", frameR)
            print "."+namefolder+"/"+namefile+"L"+str(i).zfill(2)+".jpg"
            print "."+namefolder+"/"+namefile+"R"+str(i).zfill(2)+".jpg"
    else:
        cv.ShowImage("Chessboard WebCam L", frameL)
```

```
        cv.ShowImage("Chessboard WebCam R", frameR)

    tecla = cv.WaitKey(10)
    # Se espera a la tecla ESC para salir del ciclo
    if tecla == 27:                 # Tecla ESC código ASCII
        break

# Salir
del(captureL)                       # Se cierra la conexión con la cámara
del(captureR)                       # Se cierra la conexión con la cámara
cv.DestroyAllWindows()              # Se cierra la ventana
```

A.2.2. Calibracion.py

```
#! /usr/bin/env python
# -*- encoding: utf-8 -*-

import sys
import cv2.cv as cv
import os
import glob
import time
from numpy import *


img_dir = "./imagenes/"          # Directorio donde están las imágenes del tablero de ajedrez
img_name = "a"                   # Inicio del nombre de las imágenes
file_ext = ".jpg"                # Extensión del archivo de imágenes

board_w = 7                      # Número de esquinas interiores en la horizontal del tablero de
ajedrez
board_h = 4                       # Número de esquinas interiores en la vertical del tablero de
ajedrez
board_n = board_w*board_h        # Número total de esquinas
board_sz = (board_w,board_h)     # Tamaño de la plantilla
grid_width = 2.80                 # Medida en cm del ancho de los cuadrados del tablero de
ajedrez
grid_height = 2.80                # Medida en cm del alto de los cuadrados del tablero de
ajedrez

nboard = 0                       # Número de imágenes en las que encontramos las esquinas

if __name__ == '__main__':

   # Comprobación del directorio, tiene que existir para continuar
   if os.path.isdir(img_dir) is False:
      print "Error, la carpeta: "+img_dir+" no existe"
      sys.exit(1)
   else:
```

```
    # Si existe el directorio se comprueba que hay imágenes para analizar
    nfile = len(glob.glob(img_dir + img_name + "R" + "*" + file_ext))     # Contamos el número de ficheros de imagen
    # Dimensiones de las imágenes
    imageWidth, imageHeight = cv.GetSize(cv.LoadImage(img_dir + img_name + "R" + "01" + file_ext))

    if nfile == 0:
      print "Error, la carpeta: "+img_dir+" no contine imagenes: "+img_name+'*'+file_ext
      sys.exit(1)
    else:
      cornersL = list(range(nfile))
      cornersR = list(range(nfile))
      #criteria = (cv.CV_TERMCRIT_ITER | cv.CV_TERMCRIT_EPS, 30, 0.01)

  cv.NamedWindow( "Image L", cv.CV_WINDOW_AUTOSIZE) # Ventana
  cv.NamedWindow( "Image R", cv.CV_WINDOW_AUTOSIZE) # Ventana

  # Leemos las imágenes del ajedrez
  print "Leyendo imagenes de calibracion..."
  nboard = 0
  for i in range(nfile):
    i=i+1
    filenameL = img_dir + img_name + "L" + format(i,"02d") + file_ext
    imageL = cv.LoadImage(filenameL, cv.CV_LOAD_IMAGE_GRAYSCALE)
    #cv.ShowImage("Image L", image)

    filenameR = img_dir + img_name + "R" + format(i,"02d") + file_ext
    imageR = cv.LoadImage(filenameR, cv.CV_LOAD_IMAGE_GRAYSCALE)
    #cv.ShowImage("Image R", image)

    # Find chessboard corners
    # CV_CALIB_CB_ADAPTIVE_THRESH -> indica que el umbral de intensidad es adaptativo y que irá variando según las zonas de la imagen
    # CV_CALIB_CB_NORMALIZE_IMAGE -> indica que se realizará una normalización previa de la imagen para intensificar el contraste de la imagen
```

```
      (foundL, corL) = cv.FindChessboardCorners(imageL, (board_w, board_h),
                                        cv.CV_CALIB_CB_NORMALIZE_IMAGE |
cv.CV_CALIB_CB_ADAPTIVE_THRESH)
      (foundR, corR) = cv.FindChessboardCorners(imageR, (board_w, board_h),
                                        cv.CV_CALIB_CB_NORMALIZE_IMAGE |
cv.CV_CALIB_CB_ADAPTIVE_THRESH)
      if foundL == 1 and foundR == 1:
         cornersL[nboard] = cv.FindCornerSubPix(imageL, corL, (3,3), (0,0),
(cv.CV_TERMCRIT_ITER | cv.CV_TERMCRIT_EPS, 30, 0.01))
         imageL_gray = cv.CreateMat(imageL.height, imageL.width, cv.CV_8UC3 )
         cv.CvtColor(imageL, imageL_gray, cv.CV_GRAY2RGB)
         cv.DrawChessboardCorners(imageL_gray, (board_w, board_h), cornersL[nboard],
foundL)


         cornersR[nboard] = cv.FindCornerSubPix(imageR, corR, (3,3), (0,0),
(cv.CV_TERMCRIT_ITER | cv.CV_TERMCRIT_EPS, 30, 0.01))
         imageR_gray = cv.CreateMat(imageR.height, imageR.width, cv.CV_8UC3 )
         cv.CvtColor(imageR, imageR_gray, cv.CV_GRAY2RGB)
         cv.DrawChessboardCorners(imageR_gray, (board_w, board_h), cornersR[nboard],
foundR)


         cv.ShowImage("Image L", imageL_gray)
         cv.ShowImage("Image R", imageR_gray)
         cv.WaitKey(100)


         nboard += 1
      else:
         print "No se encuentra el patron de tablero de ajedrez en ninguna imagen"



   cv.DestroyAllWindows()
   print "Numero de imagenes = ", nboard
   if nboard == 0:
      cv.DestroyAllWindows()
      print "Error, no se encuentra el patron de tablero de ajedrez en ninguna imagen"
      sys.exit(1)
```

```
# Datos para la calibración
ncor = board_w * board_h
npts = nboard * ncor
image_PointsL = cv.CreateMat(npts, 2, cv.CV_32FC1)
image_PointsR = cv.CreateMat(npts, 2, cv.CV_32FC1)
object_Points = cv.CreateMat(npts, 3, cv.CV_32FC1)
point_counts = cv.CreateMat(nboard, 1, cv.CV_32SC1)

# Calibración
intrinsic_matrix = cv.CreateMat(3, 3, cv.CV_32FC1)
distortion_coefficient = cv.CreateMat(5, 1, cv.CV_32FC1)

rvecs = cv.CreateMat(nboard, 3, cv.CV_32FC1)
tvecs = cv.CreateMat(nboard, 3, cv.CV_32FC1)

p = 0
for i in range(nboard):
   cv.SetReal2D(point_counts, i, 0, ncor)
   for j in range(ncor):
      cv.SetReal2D(object_Points, p, 0, (j / board_w) * grid_height)
      cv.SetReal2D(object_Points, p, 1, (j % board_w) * grid_width)
      cv.SetReal2D(object_Points, p, 2, 0.0)
      cv.SetReal2D(image_PointsL, p, 0, cornersL[i][j][0])
      cv.SetReal2D(image_PointsL, p, 1, cornersL[i][j][1])
      cv.SetReal2D(image_PointsR, p, 0, cornersR[i][j][0])
      cv.SetReal2D(image_PointsR, p, 1, cornersR[i][j][1])
      p += 1


# Matrices intrínsecas de cada una de las cámaras
leftIntrinsics = cv.CreateMat(3, 3, cv.CV_64F)
rightIntrinsics = cv.CreateMat(3, 3, cv.CV_64F)
# Vectores de distorsión de cada una de las cámaras
leftDistortion = cv.CreateMat(1, 5, cv.CV_64F)
rightDistortion = cv.CreateMat(1, 5, cv.CV_64F)
```

```
    # Matriz de rotación (R) y vector de translación (T)
    R = cv.CreateMat(3, 3, cv.CV_64F)
    T = cv.CreateMat(3, 1, cv.CV_64F)

    # matrices essential y fundamental
    E = cv.CreateMat(3, 3, cv.CV_64F)
    F = cv.CreateMat(3, 3, cv.CV_64F)

    cv.SetIdentity(leftIntrinsics)
    cv.SetIdentity(rightIntrinsics)
    cv.Zero(leftDistortion)
    cv.Zero(rightDistortion)

    print "calibrando..."
    # La calibración estéreo de las cámaras es un proceso iterativo, y por eso se para como
    # argumento (cv.CV_TERMCRIT_ITER+cv.CV_TERMCRIT_EPS, 30, 1e-6) que es una estructura que define
    # cuando finaliza la calibración. La operación se termina cuando se haya alcanzado el número
    # máximo de iteraciones de (30) o bien cuando el error sea menor que el mínimo (1e-6)
    #
    # CV_CALIB_FIX_PRINCIPAL_POINT

    cv.StereoCalibrate(object_Points, image_PointsL, image_PointsR, point_counts,
                       leftIntrinsics, leftDistortion, rightIntrinsics, rightDistortion,
                       (imageWidth, imageHeight), R, T, E, F,
                       (cv.CV_TERMCRIT_ITER+cv.CV_TERMCRIT_EPS, 30, 1e-6),
cv.CV_CALIB_FIX_PRINCIPAL_POINT)

    # Stereo Rectify Images
    # Matrices de rotación (R1, R2)-> corregirán las imágenes para que queden rectificadas
    R1=cv.CreateMat(3,3,cv.CV_64F)
    R2=cv.CreateMat(3,3,cv.CV_64F)
    # Matrices de proyección rectificadas (P1, P2)
    P1=cv.CreateMat(3,4,cv.CV_64F)
    P2=cv.CreateMat(3,4,cv.CV_64F)
    # Matriz que permite proyectar puntos desde una imagen al mundo real
```

```
Q=cv.CreateMat(4,4,cv.CV_64F)
print "Stereo Rectify images"
(roi1,roi2)=cv.StereoRectify(leftIntrinsics, rightIntrinsics, leftDistortion, rightDistortion,
                    (imageWidth, imageHeight), R, T, R1, R2, P1, P2, Q)

# guardamos archivos de calibración
print "Generando ficheros con todos los parametros de rectificacion"
cv.Save("r1.xml",R1)
cv.Save("r2.xml",R2)
cv.Save("p1.xml",P1)
cv.Save("p2.xml",P2)
cv.Save("q.xml",Q)
cv.Save("leftIntrinsics.xml",leftIntrinsics)
cv.Save("leftDistortion.xml",leftDistortion)
cv.Save("rightIntrinsics.xml",rightIntrinsics)
cv.Save("rightDistortion.xml",rightDistortion)

# cargamos datos de calibración
R1 = cv.Load("r1.xml")
R2 = cv.Load("r2.xml")
P1 = cv.Load("p1.xml")
P2 = cv.Load("p2.xml")
leftIntrinsics = cv.Load("leftIntrinsics.xml")
leftDistortion = cv.Load("leftDistortion.xml")
rightIntrinsics = cv.Load("rightIntrinsics.xml")
rightDistortion = cv.Load("rightDistortion.xml")

#Undistort, Rectify and Remap
#izquierda
map1x = cv.CreateMat(imageHeight, imageWidth, cv.CV_32FC1)
map2x = cv.CreateMat(imageHeight, imageWidth, cv.CV_32FC1)
#derecha
map1y = cv.CreateMat(imageHeight, imageWidth, cv.CV_32FC1)
map2y = cv.CreateMat(imageHeight, imageWidth, cv.CV_32FC1)
```

```
print "Undistort and Rectify"
# Mapa de los píxeles (mpa1x, map1y, map2x, map2y) que permiten transformar
# cualquier imagen tomada por el par de cámaras, en una imagen sin distorsión
# y rectificada.
cv.InitUndistortRectifyMap(leftIntrinsics, leftDistortion, R1, P1, map1x, map1y)
cv.InitUndistortRectifyMap(rightIntrinsics, rightDistortion, R2, P2, map2x, map2y)

print " cargando ficheros con todos los parametros de rectificacion"

# Aplicación de las transformaciones para obtener imágenes rectificadas
imageL = cv.LoadImage((img_dir + img_name + "L01" + file_ext),
cv.CV_LOAD_IMAGE_GRAYSCALE)
imageL_Remp = cv.CloneImage(imageL)

imageR = cv.LoadImage((img_dir + img_name + "R01" + file_ext),
cv.CV_LOAD_IMAGE_GRAYSCALE)
imageR_Remp = cv.CloneImage(imageR)

cv.Remap(imageL, imageL_Remp, map1x, map1y)
cv.Remap(imageR, imageR_Remp, map2x, map2y)

cv.NamedWindow( "Image Left", cv.CV_WINDOW_AUTOSIZE)
cv.NamedWindow( "Image Right", cv.CV_WINDOW_AUTOSIZE)
cv.NamedWindow( "Image Left Remp", cv.CV_WINDOW_AUTOSIZE)
cv.NamedWindow( "Image Right Remp", cv.CV_WINDOW_AUTOSIZE)

while True:
    cv.ShowImage("Image Left", imageL)
    cv.ShowImage("Image Right", imageR)
    cv.ShowImage("Image Left Remp", imageL_Remp)
    cv.ShowImage("Image Right Remp", imageR_Remp)

    tecla = cv.WaitKey(10)

    # Se espera a la tecla ESC para salir del ciclo while
    if tecla == 27:         # Tecla ESC código ASCII
        break
# Salir
#del(capture)
cv.DestroyAllWindows()
```

A.2.3. Mostrar.py

```
#! /usr/bin/env python
# -*- encoding: utf-8 -*-

import sys
import time
import cv2.cv as cv

captureWidth = 320
captureHeight = 240
cam1 = 1
cam2 = 0

if __name__ == '__main__':

   ############################################################
   # Se establece comunicación con la webcam
   ############################################################
   captureL = cv.CaptureFromCAM(cam1)
   if not captureL:                              # Control errores
      print "Error al inicializar WebCam"        # Mensaje de error
      del(captureL)                              # Cierre conexión
      sys.exit(1)                                # Salir aplicación
   # Tamaño de la ventana
   cv.SetCaptureProperty(captureL, cv.CV_CAP_PROP_FRAME_WIDTH, captureWidth)
   cv.SetCaptureProperty(captureL, cv.CV_CAP_PROP_FRAME_HEIGHT, captureHeight)

   captureR = cv.CaptureFromCAM(cam2)
   if not captureR:                              # Control errores
      print "Error al inicializar WebCam"        # Mensaje de error
      del(captureR)                              # Cierre conexión
      sys.exit(1)                                # Salir aplicación
   # Tamaño de la ventana
   cv.SetCaptureProperty(captureR, cv.CV_CAP_PROP_FRAME_WIDTH, captureWidth)
   cv.SetCaptureProperty(captureR, cv.CV_CAP_PROP_FRAME_HEIGHT, captureHeight)
   ############################################################
```

```
# cargar datos de calibración
R1 = cv.Load("xml/r1.xml")
R2 = cv.Load("xml/r2.xml")
P1 = cv.Load("xml/p1.xml")
P2 = cv.Load("xml/p2.xml")
leftIntrinsics = cv.Load("xml/leftIntrinsics.xml")
leftDistortion = cv.Load("xml/leftDistortion.xml")
rightIntrinsics = cv.Load("xml/rightIntrinsics.xml")
rightDistortion = cv.Load("xml/rightDistortion.xml")

print "Undistort and Rectify"
#Undistort, Rectify and Remap
#izquierda
map1x = cv.CreateMat(captureHeight, captureWidth, cv.CV_32FC1)
map2x = cv.CreateMat(captureHeight, captureWidth, cv.CV_32FC1)
#derecha
map1y = cv.CreateMat(captureHeight, captureWidth, cv.CV_32FC1)
map2y = cv.CreateMat(captureHeight, captureWidth, cv.CV_32FC1)

# Mapa de los píxeles (mpa1x, map1y, map2x, map2y) que permiten transformar
# cualquier imagen tomada por el par de cámaras, en una imagen sin distorsión
# y rectificada.
cv.InitUndistortRectifyMap(leftIntrinsics, leftDistortion, R1, P1, map1x, map1y)
cv.InitUndistortRectifyMap(rightIntrinsics, rightDistortion, R2, P2, map2x, map2y)

# Ventanas
cv.NamedWindow("WebCam Left", cv.CV_WINDOW_AUTOSIZE)
cv.NamedWindow("WebCam Right", cv.CV_WINDOW_AUTOSIZE)
cv.NamedWindow("WebCam Left Remap", cv.CV_WINDOW_AUTOSIZE)
cv.NamedWindow("WebCam Right Remap", cv.CV_WINDOW_AUTOSIZE)
```

```
while True:
    ####################################################
    # Se captura un frame y se muestra
    ####################################################
    frameL = cv.QueryFrame(captureL)          # Captura frame
    if frameL is None:                        # Control de errores
        print "Error captura frame Left"      # Mensaje de error
        break                                 # Salir del WHILE
    cv.ShowImage("WebCam Left", frameL)       # Muestra frame

    frameR = cv.QueryFrame(captureR)          # Captura frame
    if frameR is None:                        # Control de errores
        print "Error captura frame Right"     # Mensaje de error
        break                                 # Salir del WHILE
    cv.ShowImage("WebCam Right", frameR)      # Muestra frame
    ####################################################

    frameL_Remp = cv.CloneImage(frameL)
    frameR_Remp = cv.CloneImage(frameR)

    cv.Remap(frameL, frameL_Remp, map1x, map1y)
    cv.Remap(frameR, frameR_Remp, map2x, map2y)

    cv.ShowImage("WebCam Left Remap", frameL_Remp)
    cv.ShowImage("WebCam Right Remap", frameR_Remp)



    # Se espera a la tecla ESC para salir del ciclo
    if cv.WaitKey(10) == 27:      # Tecla ESC código ASCII
        break

# Salir
del(captureL)          # Se cierra la conexión con la cámara
del(captureR)
cv.DestroyAllWindows()  # Se cierra la ventana
```

A.2.4. Mapa.py

```
#! /usr/bin/env python
# -*- encoding: utf-8 -*-

import sys
import time
import cv2.cv as cv
from def_teclaStateSAD import *

captureWidth = 320
captureHeight = 240
cam1 = 0
cam2 = 1

if __name__ == '__main__':

    ##########################################
    # Se establece comunicación con la webcam
    ##########################################
    captureL = cv.CaptureFromCAM(cam1)
    if not captureL:                                    # Control errores
        print "Error al inicializar WebCam"             # Mensaje de error
        del(captureL)                                   # Cierre conexión
        sys.exit(1)                                     # Salir aplicación
    # Tamaño de la ventana
    cv.SetCaptureProperty(captureL, cv.CV_CAP_PROP_FRAME_WIDTH, captureWidth)
    cv.SetCaptureProperty(captureL, cv.CV_CAP_PROP_FRAME_HEIGHT, captureHeight)

    captureR = cv.CaptureFromCAM(cam2)
    if not captureR:                                    # Control errores
        print "Error al inicializar WebCam"             # Mensaje de error
        del(captureR)                                   # Cierre conexión
        sys.exit(1)                                     # Salir aplicación
    # Tamaño de la ventana
    cv.SetCaptureProperty(captureR, cv.CV_CAP_PROP_FRAME_WIDTH, captureWidth)
    cv.SetCaptureProperty(captureR, cv.CV_CAP_PROP_FRAME_HEIGHT, captureHeight)
    ##########################################
```

```
# cargar datos de calibración
R1 = cv.Load("r1.xml")
R2 = cv.Load("r2.xml")
P1 = cv.Load("p1.xml")
P2 = cv.Load("p2.xml")
leftIntrinsics = cv.Load("leftIntrinsics.xml")
leftDistortion = cv.Load("leftDistortion.xml")
rightIntrinsics = cv.Load("rightIntrinsics.xml")
rightDistortion = cv.Load("rightDistortion.xml")

print "Undistort and Rectify"
#Undistort, Rectify and Remap
#izquierda
map1x = cv.CreateMat(captureHeight, captureWidth, cv.CV_32FC1)
map2x = cv.CreateMat(captureHeight, captureWidth, cv.CV_32FC1)
#derecha
map1y = cv.CreateMat(captureHeight, captureWidth, cv.CV_32FC1)
map2y = cv.CreateMat(captureHeight, captureWidth, cv.CV_32FC1)

# Mapa de los píxeles (mpa1x, map1y, map2x, map2y) que permiten transformar
# cualquier imagen tomada por el par de cámaras, en una imagen sin distorsión
# y rectificada.
cv.InitUndistortRectifyMap(leftIntrinsics, leftDistortion, R1, P1, map1x, map1y)
cv.InitUndistortRectifyMap(rightIntrinsics, rightDistortion, R2, P2, map2x, map2y)

frameL_Remp_gray = cv.CreateImage((captureWidth, captureHeight), cv.IPL_DEPTH_8U, 1)
frameR_Remp_gray = cv.CreateImage((captureWidth, captureHeight), cv.IPL_DEPTH_8U, 1)

state = cv.CreateStereoBMState()
print "state.SADWindowSize = " + str(state.SADWindowSize)
print "state.preFilterType = " + str(state.preFilterType)
print "state.preFilterSize = " + str(state.preFilterSize)
print "state.preFilterCap = " + str(state.preFilterCap)
print "state.minDisparity = " + str(state.minDisparity)
print "state.numberOfDisparities = " + str(state.numberOfDisparities)
print "state.textureThreshold = " + str(state.textureThreshold)
```

```
print "state.uniquenessRatio = " + str(state.uniquenessRatio)
print "state.speckleRange = " + str(state.speckleRange)
print "state.speckleWindowSize = " + str(state.speckleWindowSize)

disparity = cv.CreateMat(captureHeight, captureWidth, cv.CV_32FC1)
disparity_visual = cv.CreateImage((captureWidth, captureHeight), cv.IPL_DEPTH_8U, 1)

cv.NamedWindow("WebCam Left Remap", cv.CV_WINDOW_AUTOSIZE)
cv.NamedWindow("WebCam Right Remap", cv.CV_WINDOW_AUTOSIZE)
cv.NamedWindow("Disparity Map visual", cv.CV_WINDOW_AUTOSIZE)
cv.NamedWindow("Disparity Map dist 1", cv.CV_WINDOW_AUTOSIZE)
cv.NamedWindow("Disparity Map dist 2", cv.CV_WINDOW_AUTOSIZE)
cv.NamedWindow("Disparity Map dist 3", cv.CV_WINDOW_AUTOSIZE)

aux = cv.CreateImage((captureWidth, captureHeight), cv.IPL_DEPTH_8U, 1)
disparity_dist1 = cv.CreateImage((captureWidth, captureHeight), cv.IPL_DEPTH_8U, 1)
disparity_dist2 = cv.CreateImage((captureWidth, captureHeight), cv.IPL_DEPTH_8U, 1)
disparity_dist3 = cv.CreateImage((captureWidth, captureHeight), cv.IPL_DEPTH_8U, 1)

disparity_color = cv.CreateImage((captureWidth, captureHeight), cv.IPL_DEPTH_8U, 1)

dist1_min = 221
dist1_max = 255
dist2_min = 160
dist2_max = 198
dist3_min = 98
dist3_max = 108


#cv.DrawContours
thickness = 2   #Grosor del contorno

dist1 = cv.CreateImage((captureWidth, captureHeight), cv.IPL_DEPTH_8U, 3)
dist2 = cv.CreateImage((captureWidth, captureHeight), cv.IPL_DEPTH_8U, 3)
dist3 = cv.CreateImage((captureWidth, captureHeight), cv.IPL_DEPTH_8U, 3)
```

```
    while True:
        ####################################
        # Se captura un frame y se muestra
        ####################################
        frameL = cv.QueryFrame(captureL)                    # Captura frame
        if frameL is None:                                  # Control de errores
            print "Error captura frame Left"                # Mensaje de error
            break                                           # Salir del WHILE
        #cv.ShowImage("WebCam Left", frameL)                # Muestra frame

        frameR = cv.QueryFrame(captureR)                    # Captura frame
        if frameR is None:                                  # Control de errores
            print "Error captura frame Right"               # Mensaje de error
            break                                           # Salir del WHILE
        #cv.ShowImage("WebCam Right", frameR)               # Muestra frame
        ####################################

        frameL_Remp = cv.CloneImage(frameL)
        frameR_Remp = cv.CloneImage(frameR)

        cv.Remap(frameL, frameL_Remp, map1x, map1y)
        cv.Remap(frameR, frameR_Remp, map2x, map2y)

        cv.CvtColor(frameL_Remp, frameL_Remp_gray, cv.CV_RGB2GRAY)
        cv.CvtColor(frameR_Remp, frameR_Remp_gray, cv.CV_RGB2GRAY)

        cv.Zero(disparity)
        cv.FindStereoCorrespondenceBM(frameL_Remp_gray, frameR_Remp_gray, disparity,
state)
        cv.Zero(disparity_visual)
        cv.Normalize(disparity, disparity_visual, 0, 256, cv.CV_MINMAX)

        cv.Zero(aux)
        cv.Zero(disparity_dist1)
        cv.Threshold(disparity_visual, aux, dist1_max, 255, cv.CV_THRESH_BINARY)
        cv.Threshold(disparity_visual, disparity_dist1, dist1_min, 255, cv.CV_THRESH_BINARY)
        cv.Sub(disparity_dist1, aux, disparity_dist1)
```

```
cv.Zero(aux)
cv.Zero(disparity_dist2)
cv.Threshold(disparity_visual, aux, dist2_max, 255, cv.CV_THRESH_BINARY)
cv.Threshold(disparity_visual, disparity_dist2, dist2_min, 255, cv.CV_THRESH_BINARY)
cv.Sub(disparity_dist2, aux, disparity_dist2)

cv.Zero(aux)
cv.Zero(disparity_dist3)
cv.Threshold(disparity_visual, aux, dist3_max, 255, cv.CV_THRESH_BINARY)
cv.Threshold(disparity_visual, disparity_dist3, dist3_min, 255, cv.CV_THRESH_BINARY)
cv.Sub(disparity_dist3, aux, disparity_dist3)

disparity_color = cv.CloneImage(frameL)

#cv.FindContours
storage = cv.CreateMemStorage(0)
#mode = cv.CV_RETR_LIST
mode = cv.CV_RETR_EXTERNAL
#mode = cv.CV_RETR_CCOMP
#mode = cv.CV_RETR_TREE
#method = cv.CV_CHAIN_APPROX_NONE
method = cv.CV_CHAIN_APPROX_SIMPLE
#method = cv.CV_CHAIN_APPROX_TC89_L1
#method = cv.CV_CHAIN_APPROX_TC89_KCOS

cv.Zero(dist1)
contour = cv.FindContours(disparity_dist1, storage, mode, method)
cv.DrawContours(disparity_color, contour, cv.RGB(255,0,0), cv.RGB(255,0,0), 1, thickness)
cv.DrawContours(dist1, contour, cv.RGB(255,0,0), cv.RGB(255,0,0), 1, thickness)

cv.Zero(dist2)
contour = cv.FindContours(disparity_dist2, storage, mode, method)
cv.DrawContours(disparity_color, contour, cv.RGB(0,255,0), cv.RGB(255,0,0), 1, thickness)
cv.DrawContours(dist2, contour, cv.RGB(0,255,0), cv.RGB(255,0,0), 1, thickness)
```

```
    cv.Zero(dist3)
    contour = cv.FindContours(disparity_dist3, storage, mode, method)
    cv.DrawContours(disparity_color, contour, cv.RGB(0,0,255), cv.RGB(255,0,0), 1, thickness)
    cv.DrawContours(dist3, contour, cv.RGB(0,0,255), cv.RGB(255,0,0), 1, thickness)



    # Se muestra la imagen o frame en la pantalla
    cv.ShowImage("WebCam Left Remap", frameL_Remp)
    cv.ShowImage("WebCam Right Remap", frameR_Remp)
    cv.ShowImage("Disparity Map visual", disparity_visual)
    cv.ShowImage("Disparity Map color", disparity_color)   # Muestra frame
    cv.ShowImage("Disparity Map dist 1", dist1)   # Muestra frame
    cv.ShowImage("Disparity Map dist 2", dist2)   # Muestra frame
    cv.ShowImage("Disparity Map dist 3", dist3)   # Muestra frame

    key = cv.WaitKey(10)
    keyStateSAD (key, state, DEBUG=1)
    # Se espera a la tecla ESC para salir del ciclo
    if key == 27:                           # Tecla ESC código ASCII
        break

# Salir
del(captureL)                            # Se cierra la conexión con la cámara
del(captureR)
cv.DestroyAllWindows()                   # Se cierra la ventana
```

A.2.5. Tecla.py

```
# StereoBM
def keyStateSAD (key, state, DEBUG=0):
	'''
	state.SADWindowSize -> W, w
	state.preFilterType -> F, f
	state.preFilterSize -> S, s
	state.preFilterCap -> C, c
	state.minDisparity -> M, m
	state.numberOfDisparities -> D, d
	state.textureThreshold -> T, t
	state.uniquenessRatio -> U, u
	state.speckleRange -> R, r
	state.speckleWindowSize Z, z
	'''
	# state.SADWindowSize -> W,w
	if key == 87:				#W
		if state.SADWindowSize == 255:
			state.SADWindowSize = 255
		else:
			state.SADWindowSize = state.SADWindowSize+2
		if DEBUG == 1: print "state.SADWindowSize = " + str(state.SADWindowSize)
	elif key == 119:			#w
		if state.SADWindowSize == 5:
			state.SADWindowSize = 5
		else:
			state.SADWindowSize = state.SADWindowSize-2
		if DEBUG == 1: print "state.SADWindowSize = " + str(state.SADWindowSize)

	# state.preFilterType -> F, f
	if key == 70:				#F
		if state.preFilterType == 1:
			state.preFilterType = 1
		else:
			state.preFilterType = state.preFilterType+1
```

```
        if DEBUG == 1: print "state.preFilterType = " + str(state.preFilterType)
elif key == 102:                    #f
        if state.preFilterType == 0:
                state.preFilterType = 0
        else:
                state.preFilterType = state.preFilterType-1
        if DEBUG == 1: print "state.preFilterType = " + str(state.preFilterType)

# state.preFilterSize -> S, s
if key == 83:                       #S
        if state.preFilterSize == 255:
                state.preFilterSize = 255
        else:
                state.preFilterSize = state.preFilterSize+2
        if DEBUG == 1: print "state.preFilterSize = " + str(state.preFilterSize)
elif key == 115:                    #s
        if state.preFilterSize == 5:
                state.preFilterSize = 5
        else:
                state.preFilterSize = state.preFilterSize-2
        if DEBUG == 1: print "state.preFilterSize = " + str(state.preFilterSize)

# state.preFilterCap -> C, c
if key == 67:                       #C
        if state.preFilterCap == 63:
                state.preFilterCap = 63
        else:
                state.preFilterCap = state.preFilterCap+1
        if DEBUG == 1: print "state.preFilterCap = " + str(state.preFilterCap)
elif key == 99:                     #c
        if state.preFilterCap == 1:
                state.preFilterCap = 1
        else:
                state.preFilterCap = state.preFilterCap-1
        if DEBUG == 1: print "state.preFilterCap = " + str(state.preFilterCap)
```

```
        # state.minDisparity -> M, m
        if key == 77:                   #M
                if state.minDisparity == 100:
                        state.minDisparity = 100
                else:
                        state.minDisparity = state.minDisparity+1
                if DEBUG == 1: print "state.minDisparity = " + str(state.minDisparity)
        elif key == 109:                #m
                if state.minDisparity == -100:
                        state.minDisparity = -100
                else:
                        state.minDisparity = state.minDisparity-1
                if DEBUG == 1: print "state.minDisparity = " + str(state.minDisparity)

        # state.numberOfDisparities -> D, d
        if key == 68:                   #D
                if state.numberOfDisparities == 255:
                        state.numberOfDisparities = 255
                else:
                        state.numberOfDisparities = state.numberOfDisparities+16
                if DEBUG == 1: print "state.numberOfDisparities = " +
str(state.numberOfDisparities)
        elif key == 100:                #d
                if state.numberOfDisparities == 16:
                        state.numberOfDisparities = 16
                else:
                        state.numberOfDisparities = state.numberOfDisparities-16
                if DEBUG == 1: print "state.numberOfDisparities = " +
str(state.numberOfDisparities)

        # state.textureThreshold -> T, t
        if key == 84:                   #T
                if state.textureThreshold == 255:
                        state.textureThreshold = 255
                else:
                        state.textureThreshold = state.textureThreshold+1
                if DEBUG == 1: print "state.textureThreshold = " + str(state.textureThreshold)
```

```
elif key == 116:                    #t
        if state.textureThreshold == 0:
                state.textureThreshold = 0
        else:
                state.textureThreshold = state.textureThreshold-1
        if DEBUG == 1: print "state.textureThreshold = " + str(state.textureThreshold)

# state.uniquenessRatio -> U, u
if key == 85:                       #U
        if state.uniquenessRatio == 255:
                state.uniquenessRatio = 255
        else:
                state.uniquenessRatio = state.uniquenessRatio+1
        if DEBUG == 1: print "state.uniquenessRatio = " + str(state.uniquenessRatio)
elif key == 117:                    #u
        if state.uniquenessRatio == 0:
                state.uniquenessRatio = 0
        else:
                state.uniquenessRatio = state.uniquenessRatio-1
        if DEBUG == 1: print "state.uniquenessRatio = " + str(state.uniquenessRatio)

# state.speckleRange -> R, r
if key == 82:                       #R
        if state.speckleRange == 255:
                state.speckleRange = 255
        else:
                state.speckleRange = state.speckleRange+1
        if DEBUG == 1: print "state.speckleRange = " + str(state.speckleRange)
elif key == 114:                    #r
        if state.speckleRange == 0:
                state.speckleRange = 0
        else:
                state.speckleRange = state.speckleRange-1
        if DEBUG == 1: print "state.speckleRange = " + str(state.speckleRange)
```

```
        # state.speckleWindowSize Z, z
        if key == 90:                           #Z
                if state.speckleWindowSize == 255:
                        state.speckleWindowSize = 255
                else:
                        state.speckleWindowSize = state.speckleWindowSize+1
                if DEBUG == 1: print "state.speckleWindowSize = " +
str(state.speckleWindowSize)
        elif key == 122:                        #z
                if state.speckleWindowSize == 0:
                        state.speckleWindowSize = 0
                else:
                        state.speckleWindowSize = state.speckleWindowSize-1
                if DEBUG == 1: print "state.speckleWindowSize = " +
str(state.speckleWindowSize)
```